Министерство образования и науки РФ

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**

**МЕТОДЫ СОРТИРОВКИ**

Выполнил: Шагов Максим Алексеевич, студент группы 381806-1

Проверил: к. т. н., Доцент кафедры МОСТ Кустикова В. Д.

Нижний Новгород

2018

Содержание

[Введение 3](#_Toc533274910)

[1. Постановка задачи 4](#_Toc533274911)

[2. Руководство пользователя 5](#_Toc533274912)

[3. Руководство программиста 7](#_Toc533274913)

[3.1. Структура программы 7](#_Toc533274914)

[3.2. Описание алгоритма 7](#_Toc533274915)

[3.2.1. Сортировка выбором 7](#_Toc533274916)

[3.2.2. Сортировка простыми вставками 7](#_Toc533274917)

[3.2.3. Пузырьковая сортировка 8](#_Toc533274918)

[3.2.4. Сортировка подсчетом 10](#_Toc533274919)

[3.2.5. Быстрая сортировка 10](#_Toc533274920)

[3.2.6. Сортировка слиянием 11](#_Toc533274921)

[3.3. Описание функций 13](#_Toc533274922)

[Заключение 15](#_Toc533274923)

# Введение

На данный момент все без исключения пользователям персональных компьютеров приходится работать с директориями и файлами, находящимся в них. Целесообразно упростить работу с файловой системой. Поэтому и возникает задача - создания некоторой программы упрощающей взаимодействие с системой.

Для многих людей удобнее рассматривать файлы в отсортированном виде. Например, по возрастанию размеров файлов. Таким образом, для этой цели и был создан прототип файлового менеджера, с помощью которого можно легко вывести на экран отсортированный одним из 6 способов набор файлов с указанием размеров.

# Постановка задачи

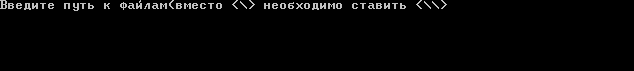
Задача: Разработать программу «Файловый менеджер», способный сортировать файлы несколькими и выводить время выполнения сортировки.

Входные данные: Путь к файлам.

**Выходные данные**: Отсортированный список файлов и время, за которое были отсортированы файлы.

# Руководство пользователя

1. Запустить файл **Practice5.exe.** В появившимся окне ввести путь до файлов. (см. Рис 1)



1. Запуск программы Practice5.exe.
2. Следуя указаниям программы, вводим путь с клавиатуры. Обращаю внимание, что все символы “**\**” нужно заменить на “**\\**”

Например, Изначальный путь к файлам: C:\Users\user\\Desktop\files.

После замены: C:\\Users\\user\\Desktop\\files

Вид программ при этом действии. (см. Рис 2) .

Если в программа не найдет файлов в директории, то оповестит пользователя об этом. (см. Рис 3)

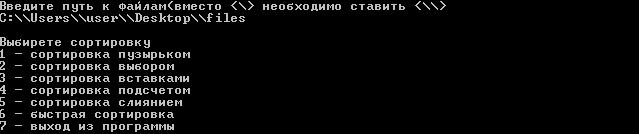
Если путь введен неправильно, то программа закроется автоматически.



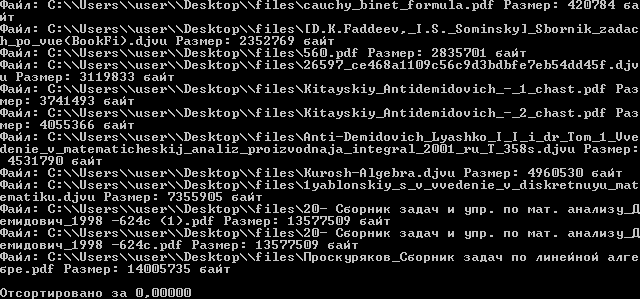
1. Ввод пути.



1. Сообщение о том, что папка пустая
2. Нажимаем на кнопку Enter. На экране появиться следующее меню. (см. Рис 4)



1. Вывод меню.
2. Пользователь выбирает сортировку и вводит ее номер. После этого нажимает клавишу Enter (см .Рис 5)



1. Вывод отсортированного списка файлов.

**Примечания**: Для сортировки номер 4 (Подсчетом), если размер файлов находятся в слишком большом диапазоне, то сортировать таким образом невозможно. Выводится сообщение об этом

# Руководство программиста

## Структура программы

Practice5.c – исходный код программы.

## Описание алгоритма

Список файлов находящийся по введенному пути сортируется одним из 6 способов.

### Сортировка выбором

Описание: алгоритм по сортировке массива. Сложность сортировки О(N2). Память: О(1).

Алгоритм:

Находим номер минимального элемента в текущем списке.

Производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент находится на данной позиции)

Теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы.

Пример: Таблица 1. Исходный массив: 8 3 4 2 6 7 4

1. Пример сортировки выбором.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | Массив элементов | | | | | | | Словестное описание шагов |
| 1 | 8 | 3 | 4 | 2 | 6 | 7 | 4 | Ввод изначального массива |
| 2 | 2 | 3 | 4 | 8 | 6 | 7 | 4 | Минимальный элемент 2. Меняем его местами с нулевым элементом. |
| 3 | 2 | 3 | 4 | 8 | 6 | 7 | 4 | Минимальный элемент 3. Меняем его местами с 1 элементом. |
| 4 | 2 | 3 | 4 | 8 | 6 | 7 | 4 | Минимальный элемент 4. Он находится на своей позиции |
| 5 | 2 | 3 | 4 | 4 | 6 | 7 | 8 | Минимальный элемент 4. Меняем его местами с 4 элементом. |
| 6 | 2 | 3 | 4 | 4 | 6 | 7 | 8 | Минимальный элемент 6. Он находится на своей позиции |
| 7 | 2 | 3 | 4 | 4 | 6 | 7 | 8 | Минимальный элемент 7. Он находится на своей позиции |
| 8 | 2 | 3 | 4 | 4 | 6 | 7 | 8 | Остался последний элемент он уже стоит на своем месте. |

Примечание к таблице. Цветом выделена уже отсортированная часть массива

Отсортированный массив: 2 3 4 4 6 7 8

### Сортировка простыми вставками

Описание: алгоритм по сортировке массива c помощью вставки в уже отсортированный фрагмент массива. Сложность сортировки: О(N2). Память: О(1).

Алгоритм:

Выбираем первый элемент и считаем его отсортированным.

Берем следующий элемент массива, сравниваем его с последним отсортированным, если сравниваемый элемент больше, то меняем их местами.

Повторяем эти действия пока не встретиться число меньшее или не достигнем левого края массива.

Повторяем действия 1 – 3 пока массив не будет отсортирован.

Пример: Таблица 2. Исходный массив: 8 3 4 2 6 7 4

1. Пример сортировки простыми вставками.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | Массив элементов | | | | | | | Словестное описание шагов |
| 1 | 8 | 3 | 4 | 2 | 6 | 7 | 4 | Ввод изначального массива |
| 2 | 8 | 3 | 4 | 2 | 6 | 7 | 4 | Первый элемент считаем отсортированным |
| 3 | 3 | 8 | 4 | 2 | 6 | 7 | 4 | Берем элемент 3. Добавляем его в отсортированную часть массива на свое место. |
| 4 | 3 | 4 | 8 | 2 | 6 | 7 | 4 | Берем элемент 4. Добавляем его в отсортированную часть массива на свое место. |
| 5 | 2 | 3 | 4 | 8 | 6 | 7 | 4 | Берем элемент 2. Добавляем его в отсортированную часть массива на свое место. |
| 6 | 2 | 3 | 4 | 6 | 8 | 7 | 4 | Берем элемент 6. Добавляем его в отсортированную часть массива на свое место. |
| 7 | 2 | 3 | 4 | 6 | 7 | 8 | 4 | Берем элемент 7. Добавляем его в отсортированную часть массива на свое место. |
| 8 | 2 | 3 | 4 | 4 | 6 | 7 | 8 | Берем элемент 4. Добавляем его в отсортированную часть массива на свое место. |

Отсортированный массив: 2 3 4 4 6 7 8

### Пузырьковая сортировка

Описание: алгоритм по сортировке массива с помощью «Пузырька». Сложность сортировки: О(N2). Память: О(1).

Алгоритм:

1. Сравниваем нулевой и первый элемент, если первый элемент меньше, то меняем местами их.
2. Сравниваем первый и второй элемент, если второй элемент меньше, то меняем местами их.

На некотором шаге дойдем до конца массива, получив в последней позиции наибольший элемент. Проходы по массиву повторяются (N – 1) раз до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован.

Пример: Таблица 3. Исходный массив: 8 3 4 2 6 7 4

1. Пример 1 прохода пузырьковой сортировки.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | Массив элементов | | | | | | | Словестное описание шагов |
| 1 | 8 | 3 | 4 | 2 | 6 | 7 | 4 | Ввод изначального массива |
| 2 | 3 | 8 | 4 | 2 | 6 | 7 | 4 | Сравниваем нулевой и первый элемент. Меняем местами, если второй элемент меньше |
| 3 | 3 | 4 | 8 | 2 | 6 | 7 | 4 | Сравниваем 1 и 2 элемент. Меняем местами, если 2 элемент меньше |
| 4 | 3 | 4 | 2 | 8 | 6 | 7 | 4 | Сравниваем 2 и 3 элемент. Меняем местами, если 3 элемент меньше |
| 5 | 3 | 4 | 2 | 6 | 8 | 7 | 4 | Сравниваем 3 и 4 элемент. Меняем местами, если 4 элемент меньше |
| 6 | 3 | 4 | 2 | 6 | 7 | 8 | 4 | Сравниваем 4 и 5 элемент. Меняем местами, если 5 элемент меньше |
| 7 | 3 | 4 | 2 | 6 | 7 | 4 | 8 | Сравниваем 5 и 6 элемент. Меняем местами, если 6 элемент меньше |

Был показан первый проход. Приведу результат в таблице после N – 1 прохода. (см Таблица 4).

1. Результат работы после N – 1 прохода.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №прохода | Массив элементов | | | | | | | Словестное описание шагов |
| 1 | 3 | 4 | 2 | 6 | 7 | 4 | 8 | Наибольший элемент 8 |
| 2 | 3 | 2 | 4 | 6 | 4 | 7 | 8 | Наибольший элемент 7( в не отсортированной части) |
| 3 | 3 | 2 | 4 | 4 | 6 | 7 | 8 | Наибольший элемент 6( в не отсортированной части) |
| 4 | 2 | 3 | 4 | 4 | 6 | 7 | 8 | Массив отсортирован |

Проходов получилось меньше, чем N – 1. Т.к. был рассмотрен не самый худший вариант сортировки.

Отсортированный массив: 2 3 4 4 6 7 8

### Сортировка подсчетом

Описание: алгоритм по сортировке массива целых чисел в диапазоне k (k = Максимальный элемент – минимальный элемент + 1), работающий за линейное время. Сложность сортировки: O(N). Память: дополнительный массив из k элементов, где k - диапазон чисел.

Алгоритм:

Создаем дополнительный массив из k элементов (Массив посчёта) и всем элементам присваиваем ноль.

Проходим по изначальному массиву и увеличиваем значения i–ых элементов массива подсчета на единицу, каждый раз встречая их в изначальном.

Записываем в массив число i столько раз, сколько в массиве посчета на i-той позиции. ( i от 0 до k – 1)

Таблица 5. Исходный массив: 8 3 4 2 6 7 4

1. Пример сортировки подсчетом.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Индексы | | | | | | |
| Номер элемента в массиве | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Массив |  | | | | | | |
| Изначальный | 8 | 3 | 4 | 2 | 6 | 7 | 4 |
| Подсчета | 1 | 1 | 2 | 1 | 1 | 1 |  |
| Изначальный(отсортированный) | 2 | 3 | 4 | 4 | 6 | 7 | 8 |

Отсортированный массив: 2 3 4 4 6 7 8

### Быстрая сортировка

Описание: рекурсивный алгоритм, выполняющий сортировку массива. Сложность сортировки: О(N\*log2N). Память: О(1).

Алгоритм:

Выбираем средний элемент и делим массив на две части.

Перемещаем все элементы больше среднего справа от него, а все элементы меньше влево.

Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

Массив считается отсортированным, когда в каждой части останется по 1 элементу.

Пример: Таблица 6. Исходный массив: 9 8 6 4 2 1 1

1. Пример быстрой сортировки

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | Массив элементов | | | | | | | Словестное описание шагов |
| 1 | 9 | 8 | 6 | 4 | 3 | 1 | 1 | Ввод изначального массива |
| 2 | 1 | 1 | 3 | 4 | 9 | 8 | 6 | Средний элемент 4 Меняем местами большие и меньшие элементы |
| 3 | 1 | 1 | 3 | 4 | 9 | 8 | 6 | Рассматриваем элементы большей и меньшей части. Заново выбираем элемент середины для каждой  В меньшей: 1  В большей: 8 |
| 4 | 1 | 1 | 3 | 4 | 6 | 8 | 9 | Рассматриваем элементы большей и меньшей части (для каждой из частей). Заново выбираем элемент середины для каждой. |
| 5 | 1 | 1 | 3 | 4 | 6 | 8 | 9 | Оставшиеся разбиты на массивы по одному элементу. Значит массив отсортирован. |

Отсортированный массив: 1 1 3 4 6 8 9

### Сортировка слиянием

Описание: рекурсивный алгоритм, реализующий сортировку по принципу слияния двух частей массива. Сложность сортировки: О(N\*log2N). Память: О(N).

Алгоритм:

1. Сортируемый массив разбивается на две части примерно одинакового размера;
2. Каждая из получившихся частей сортируется отдельно, тем же самым алгоритмом;
3. Два упорядоченных массива половинного размера соединяются в один. А именно поочередно сравнивают по два элемента разных частей массива и записывают наименьший из их. Если один из массивов закончился, то в хвост записывают оставшиеся значения из другой части.

Пример: Таблица 7. Исходный массив: 8 3 4 2 6 7 4

1. Пример сортировки слиянием

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | Массив | | | | | | |  |
| 1 | 8 | 3 | 4 | 2 | 6 | 7 | 4 | Ввод массива |
| 2 | 8 | 3 | 4 | 2 | 6 | 7 | 4 | Разбили на две части |
| 3 | 8 | 3 | 4 | 2 | 6 | 7 | 4 | Разбили каждую часть еще на две части |
| 4 | 8 | 3 | 4 | 2 | 6 | 7 | 4 | Остались массивы по одному элементу. Начинаем сливать их |
| 5 | 3 | 8 | 2 | 4 | 6 | 7 | 4 |  |
| 6 | 2 | 3 | 4 | 8 | 4 | 6 | 7 |  |
| 7 | 2 | 3 | 4 | 4 | 6 | 7 | 8 | Окончательное слияние |

Отсортированный массив: 2 3 4 4 6 7 8

## Описание функций

void pick(ULONGLONG a[], int \*b, int n)

**Назначение**: реализует сортировку выбором для массива.

**Входные параметры:** а – массив с основными значениями.

b – массив индексов.

n – длина массива.

**Выходные данные**: отсутствуют.

void paste(ULONGLONG a[], int n, int \*b)

**Назначение**: реализует сортировку вставками для массива.

**Входные параметры**: а – массив с основными значениями.

b – массив индексов.

n – длина массива.

**Выходные данные**: отсутствуют.

void bubble\_sort(ULONGLONG a[],int \*b, int n)

**Назначение**: реализует сортировку пузырьком для массива.

**Входные параметры**: а – массив с основными значениями.

b – массив индексов.

n – длина массива.

**Выходные данные**: отсутствуют.

void counting\_sort(ULONGLONG a[], int \*b, int n)

**Назначение**: реализует сортировку подсчетом для массива.

**Входные параметры**: а – массив с основными значениями.

b – массив индексов.

n – длина массива.

**Выходные данные**: отсутствуют.

void merge(ULONGLONG a[], int \*b, int l, int m, int r)

**Назначение**: реализует слияние 2 частей массива.

**Входные параметры**: а – массив с основными значениями.

b – массив индексов.

l – левая граница массива.

r - правая граница массива.

m- выбранная середина массива

**Выходные данные**: отсутствуют.

void merge\_sort(ULONGLONG a[], int l, int r, int \*b)

**Назначение**: реализует сортировку слиянием для массива.

**Входные параметры**: а – массив с основными значениями.

b – массив индексов.

l – левая граница массива.

r - правая граница массива.

**Выходные данные**: отсутствуют.

void hoarasort(ULONGLONG\* a, int \*b, int first, int last)

**Назначение**: реализует сортировку выбором для массива.

**Входные параметры**: а – массив с основными значениями.

b – массив индексов.

first – индекс первого элемента.

last – индекс последнего элемента.

**Выходные данные**: отсутствуют.

int ListDirectoryContents(wchar\_t\*sDir, wchar\_t\*\*\*filename, ULONGLONG \*\*filesize)

**Назначение**: реализует запись имен файлов и их размеров в массивы и считает количество файлов в директории.

**Входные параметры**: sDir – строка, содержащая путь к директории.

filename – массив слов.

filesize – массив размеров файлов.

**Выходные данные**: количество файлов в директории.

void Putpath(wchar\_t\* str, char\* str1)

**Назначение**: реализует ввод пути и преобразует строку типа char в строку типа wchar\_t.

**Входные параметры**: str – строка типа wchar\_t.

str1 – строка типа char.

**Выходные данные**: отсутствуют.

void call\_menu()

**Назначение**: вызов меня выбора.

**Входные параметры**: отсутствуют

**Выходные данные**: отсутствуют.

# Заключение

В ходе выполнения работы получена программа «Файлового менеджера». Она способна, считывая путь к директории, выводить отсортированный массив файлов по возрастанию их размеров. Использованные сортировки: сортировка выбором, сортировка простыми вставками, пузырьковая сортировка, сортировка подсчетом, быстрая сортировка, сортировка слиянием.

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) |