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

Макарычев С.Д.

Проверил:

Доцент кафедры МОСТ ИИТММ

Кустикова В.Д.

Нижний Новгород

2018 г.

**Содержание**

[Введение 4](#_Toc533596309)

[1 Постановка задач 5](#_Toc533596310)

[2 Руководство пользователя 6](#_Toc533596311)

[3 Руководство программиста 7](#_Toc533596312)

[3.1 Структура программы 7](#_Toc533596313)

[3.2 Описание алгоритмов 7](#_Toc533596314)

[3.2.1 Сортировка выбором 7](#_Toc533596315)

[3.2.2 Сортировка вставками 7](#_Toc533596316)

[3.2.3 Сортировка пузырьком 8](#_Toc533596317)

[3.2.4 Сортировка подсчётом 9](#_Toc533596318)

[3.2.5 Сортировка слиянием 9](#_Toc533596319)

[3.2.6 Сортировка Хоара 10](#_Toc533596320)

[4 Описание функций 11](#_Toc533596321)

[Заключение 13](#_Toc533596322)

[Литература 14](#_Toc533596323)

# Введение

В настоящее время одной из важнейших операций, применяемых пользователями ПК, является сортировка массивов по определённому критерию. Это позволяет значительно снизить затраты времени при поиске определённого элемента массива.

В данной работе представлен прототип файлового менеджера, позволяющего отсортировать файлы в определённой директории по размеру одним из шести способов сортировки.

# Постановка задач

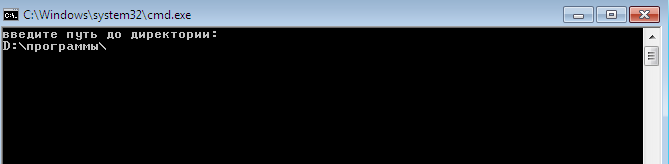
В данной работе необходимо реализовать программу прототип файлового менеджера, сортирующего файлы в заданной директории по размеру одним из шести методов: выбором, вставками, пузырьковым, подсчётом, слиянием, Хоара.

Входные данные: путь до нужной директории, выбор метода сортировки.

Выходные данные: отсортированный список имён и размеров файлов, время сортировки.

# Руководство пользователя

После запуска программы пользователю нужно ввести путь до директории, в которой нужно осуществить сортировку. Например, нужно отсортировать файлы в директории <D:\программы\>. Вводим в поле данный путь и нажимаем Enter.(рис.1)

****

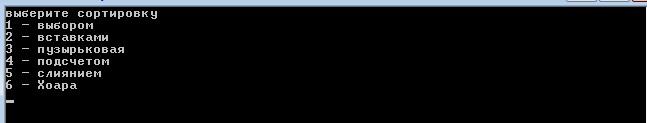
1. Ввод пути

После ввода на экране появится заданный каталог.(рис.2)



1. Вывод файлов из заданного каталога

После этого программа предложит вам выбрать метод сортировки.(рис.3)



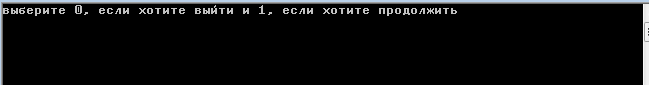
1. Выбор способа сортировки

После выбора сортировки программа выводит отсортированные выбранным способом каталог и время сортировки.(рис.4)



1. Вывод отсортированного каталога и времени сортировки

Затем программа предложит вам продолжить работать с каталогами или выйти. Если хотите остаться, введите “1”. Если хотите выйти введите “0”.(рис.5)



1. Вывод дальнейшего сценария программы

Если вы захотите остаться, алгоритм начнёт выполняться заново.

# Руководство программиста

## Структура программы

Данная программа состоит из одного файла main.c, в котором находится весь её код.

## Описание алгоритмов

### Сортировка выбором

В его основе лежит операция сравнения. В неотсортированном подмассиве ищется локальный максимум (минимум). Найденный максимум (минимум) меняется местами с последним (первым) элементом в подмассиве. Если в массиве остались неотсортированные подмассивы — начинаем заново искать локальный максимум (минимум) и менять местами с последним (первым) элементом в подмассиве. И.т.д.

Пусть нам дан неотсортированный массив из n элементов. Будем строить готовую последовательность, начиная с левого конца массива от нулевого элемента и заканчивая n– 1.

1. Пример сортировки выбором

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

### Сортировка вставками

Общая суть сортировок вставками такова: перебираются элементы в неотсортированной части массива; каждый элемент вставляется в отсортированную часть массива на то место, где он должен находиться.

1. Пример сортировки вставками

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Индекс  Шаг | Массив | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | ***6*** | 9 | 3 | -5 | 8 | 2 | -4 |
| 2 | ***6*** | ***9*** | 1 | -5 | 8 | 2 | -4 |
| 3 | ***6*** | ***9*** | ***1*** | -5 | 8 | 2 | -4 |
| *6* | *1* | 9 | -5 | 8 | 2 | -4 |
| *1* | *6* | *9* | -5 | 8 | 2 | -4 |
| 4 | ***1*** | ***6*** | ***9*** | ***-5*** | 8 | 2 | -4 |
| *1* | *6* | *-5* | *9* | 8 | 2 | -4 |
| *1* | *-5* | *6* | *9* | 8 | 2 | -4 |
| *-5* | *1* | *6* | *9* | 8 | 2 | -4 |
| 5 | ***-5*** | ***1*** | ***6*** | ***9*** | ***8*** | 2 | -4 |
| *-5* | *1* | *6* | *8* | *9* | 2 | -4 |
| 6 | ***-5*** | ***1*** | ***6*** | ***8*** | ***9*** | ***2*** | -4 |
| *-5* | *1* | *6* | *8* | *2* | 9 | *-4* |
| *-5* | *1* | *6* | *2* | 8 | *9* | *-4* |
| *-5* | *1* | *2* | *6* | 8 | *9* | *-4* |
| 7 | ***-5*** | ***1*** | ***2*** | ***6*** | ***8*** | ***9*** | ***-4*** |
| *-5* | *1* | *2* | *6* | 8 | *-4* | *9* |
| *-5* | *1* | *2* | *6* | -4 | *8* | *9* |
| *-5* | *1* | *2* | *-4* | 6 | *8* | *9* |
| *-5* | *1* | *-4* | *2* | 6 | *8* | *9* |
| *-5* | *-4* | *1* | *2* | 6 | *8* | *9* |
| *-4* | *-2* | *1* | *2* | 6 | *8* | *9* |
| Отсортированный массив | *-4* | *-2* | *1* | *2* | 6 | *8* | *9* |

### Сортировка пузырьком

Идём по массиву чисел и проверяем порядок (следующее число должно быть больше и равно предыдущему), как только наткнулись на нарушение порядка, тут же обмениваем местами элементы, доходим до конца массива, после чего начинаем сначала.

1. Пример сортировки пузырьком

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Индекс  Шаг | Массив | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 6 | 9 | 3 | -5 | 8 | 2 | -4 |
| 2 | 6 | 9 | 3 | -5 | 8 | 2 | -4 |
| 6 | 3 | 9 | -5 | 8 | 2 | -4 |
| 6 | 3 | -5 | 9 | 8 | 2 | -4 |
| 6 | 3 | -5 | 8 | 9 | 2 | -4 |
| 6 | 3 | -5 | 8 | 2 | 9 | -4 |
| 6 | 3 | -5 | 8 | 2 | -4 | 9 |
| 3 | 6 | -5 | 3 | 8 | 2 | -4 | 9 |
| 6 | -5 | 3 | 2 | 8 | -4 | 9 |
| 6 | -5 | 3 | 2 | -4 | 8 | 9 |
| 3 | -5 | 6 | 3 | 2 | -4 | 8 | 9 |
| -5 | 3 | 6 | 2 | -4 | 8 | 9 |
| -5 | 3 | 2 | 6 | -4 | 8 | 9 |
| -5 | 3 | 2 | -4 | 6 | 8 | 9 |
| 4 | -5 | 2 | 3 | -4 | 6 | 8 | 9 |
| -5 | 2 | -4 | 3 | 6 | 8 | 9 |
| 5 | -5 | -4 | 2 | 3 | 6 | 8 | 9 |
| Отсортированный массив | -5 | -4 | 2 | 3 | 6 | 8 | 9 |

### Сортировка подсчётом

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 подсчётом

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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

Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи.

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 |

### Сортировка Хоара

Отличительной особенностью быстрой сортировки является операция разбиения массива на две части относительно опорного элемента. Например, если последовательность требуется упорядочить по возрастанию, то в левую часть будут помещены все элементы, значения которых меньше значения опорного элемента, а в правую элементы, чьи значения больше или равны опорному. Вне зависимости от того, какой элемент выбран в качестве опорного, массив будет отсортирован, но все же наиболее удачным считается ситуация, когда по обеим сторонам от опорного элемента оказывается примерно равное количество элементов. Если длина какой-то из получившихся в результате разбиения частей превышает один элемент, то для нее нужно рекурсивно выполнить упорядочивание, т. е. повторно запустить алгоритм на каждом из отрезков.

1. Пример сортировки Хоара

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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

**Функция «Choose»**

Описание: реализация сортировки выбором.

Принимает на вход количество элементов в директории n и массив структур b[].

В результате выполнения функции, массив b становится упорядоченным.

**Функция «Insert»**

Описание: реализация сортировки вставками.

Принимает на вход количество элементов в директории n и массив структур b[]. В результате выполнения функции, массив b становится упорядоченным.

**Функция «Bubble»**

Описание: реализация сортировки пузырьком.

Принимает на вход количество элементов в директории n и массив структур b[]. В результате выполнения функции, массив b становится упорядоченным.

**Функция «Count\_sort»**

Описание: сортировка подсчетом

Принимает на вход количество элементов в директории n, массив структур b[] и количество значений k. В результате выполнения функции, массив b становится упорядоченным.

**Функция «Merge\_sort»**

Описание: реализация сортировки слиянием.

Принимает на вход массив структур b[], индекс первого и последнего элементов..

**Функция «Merge»**

Описание: реализует слияние двух упорядоченных массивов.

Принимает на вход массив b, индекс левой границы , индекс середины m, индекс правой границы r.. В результате получается слитый массив a.

**Функция «Quick\_sort»**

Описание: реализация сортировки Хоара.

Принимает на вход массив структур b[], индекс первого и последнего элементов. В результате выполнения функции, массив b становится упорядоченным.

**Функция «sum\_files»**

Описание:нахождение количества файлов в директории

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

Выходные параметры: количество файлов в директории

**Функция «ListDirectoryContents»**

Описание:вывод каталога на экран, формирование и заполнение массива структур

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

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

**Функция «print»**

Описание:вывод отсортированного каталога и времени сортировки на экран,

Принимает на вход количество элементов в директории n, массив структур b[] и время работы функций.

# Заключение

Чтобы определить, какая сортировка является наиболее быстрой, был проведён эксперимент, в котором измерялось время, за которое каждая сортировка упорядочивает файлы по размеру. В таблице приведены сравнение времени работы и значения памяти, необходимые для разных видов сортировок.

1. Эффективность сортировок

|  |  |  |
| --- | --- | --- |
| **Методы сортировки** | **Время выполнения** | **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 |
| Быстрая сортировка | O(N·log2 N) | O(1) |
| Сортировка слиянием | O(N·log2 N) | O(N) |

# Литература

1. Сайт habr. Сортировка выбором - <https://habr.com/post/422085/>
2. Сайт habr. Сортировка вставками - <https://habr.com/post/415935/>
3. Сайт learnc. Сортировка пузырьком - <https://learnc.info/algorithms/bubblesort.html>
4. Сайт vscode Сортировка подсчетом - <https://vscode.ru/prog-lessons/sortirovka-metodom-podscheta-na-si.html>
5. Сайт forkettle Сортировка слиянием - <https://forkettle.ru/vidioteka/programmirovanie-i-set/algoritmy-i-struktury-dannykh/108-sortirovka-i-poisk-dlya-chajnikov/1015-sortirovka-sliyaniem-merge-sort>
6. Сайт forkettle Сортировка Хоара - https://forkettle.ru/vidioteka/programmirovanie-i-set/108-algoritmy-i-struktury-dannykh/sortirovka-i-poisk-dlya-chajnikov/1010-metod-khoara-bystraya-sortirovka-quick-sort