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: учащийся группы 381806-1 Тактаев А.А.

Проверил: доцент кафедры МОСТ ИИТММ Кустикова В.Д.

Нижний Новгород 2018 г.

Оглавление

[1 Введение. 3](#_Toc533367833)

[2 Постановка задачи. 4](#_Toc533367834)

[3 Руководство пользователя. 5](#_Toc533367835)

[4 Руководство программиста. 8](#_Toc533367836)

[4.1 Структура программы. 8](#_Toc533367837)

[4.2 Описание алгоритмов. 8](#_Toc533367838)

[4.2.1 Сортировка выбором. 8](#_Toc533367839)

[4.2.2 Сортировка простыми вставками. 8](#_Toc533367840)

[4.2.3 Сортировка пузырьком. 9](#_Toc533367841)

[4.2.4 Сортировка подсчетом. 9](#_Toc533367842)

[4.2.5 Быстрая сортировка. 10](#_Toc533367843)

[4.2.6 Сортировка слиянием. 10](#_Toc533367844)

[4.3 Описание фукций 11](#_Toc533367845)

[5 Заключение. 13](#_Toc533367846)

# Введение.

Каждый человек на протяжении всей своей жизни, сам того не замечая, обрабатывает огромные объемы информации. С появлением компьютера стала возможна обработка еще больших количеств этой самой информации.

Но и человеку, и компьютеру гораздо удобнее работать с отсортированным по какому-либо критерию списком данных. Из этих соображений человек придумал алгоритмы сортировки информации. Некоторые из этих сортировок и показаны в данной работе на примере файлового менеджера, сортирующего файлы по их размеру.

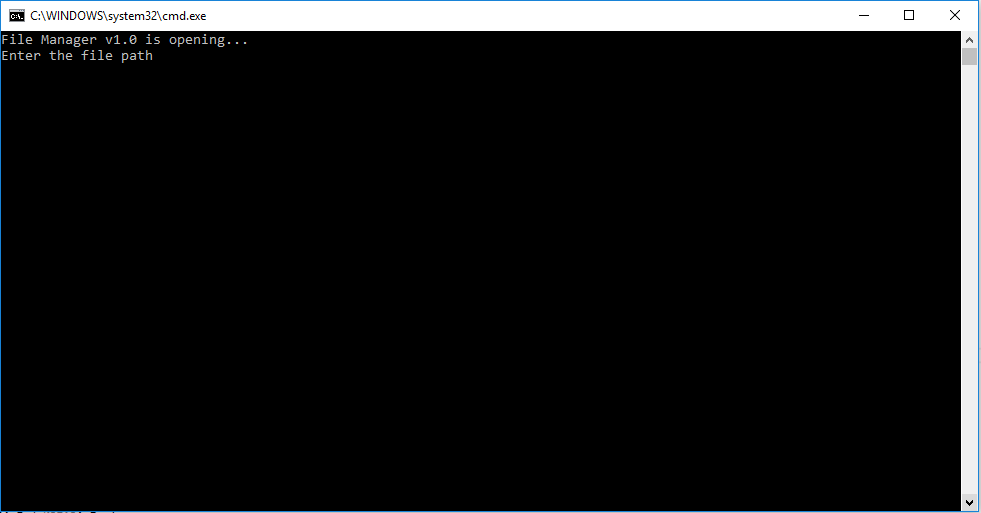
# Постановка задачи.

Реализовать программу-прототип файлового менеджера на языке программирования C, который сортирует выбранным способом файлы по введенному пути по возрастанию/убыванию их размера.

# Руководство пользователя.

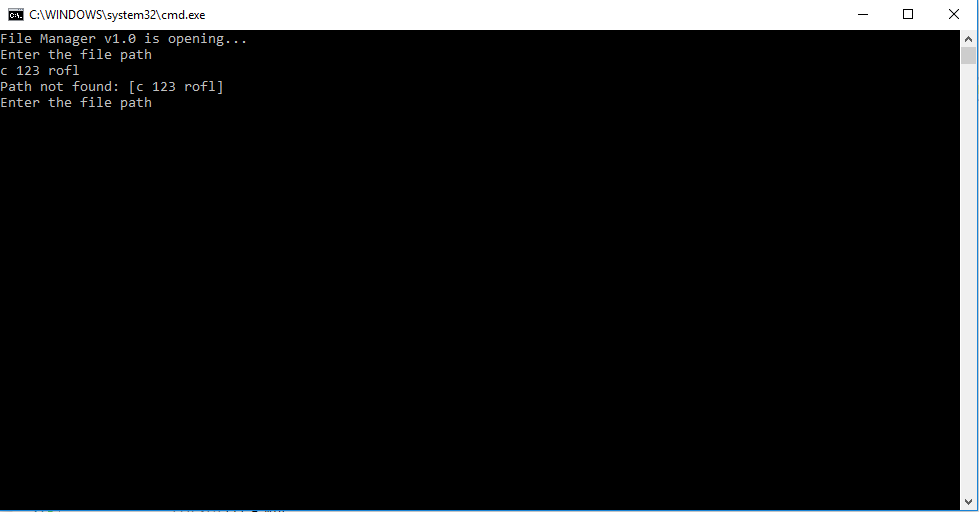
Здесь содержится подробное описание действий, приводящих к успешной работе программы:

Запустить файл под названием **Practice5.exe**. На экране появится сообщение с предложением ввести путь к какой-либо папке (Рис.1).



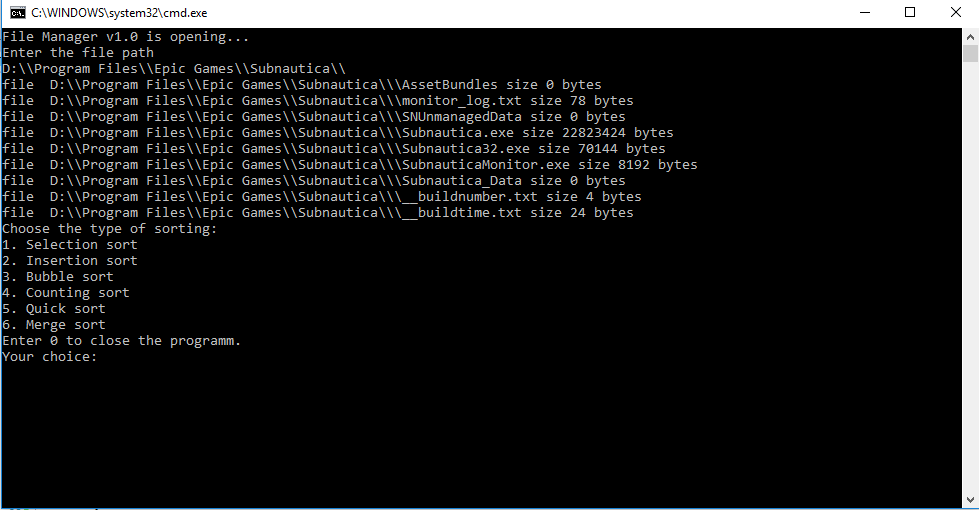
1. Запуск программы.

Ввести путь к какой-либо папке на вашем компьютере и нажмите Enter. Если ввести несуществующий путь, то программа выдаст ошибку (Рис.2).

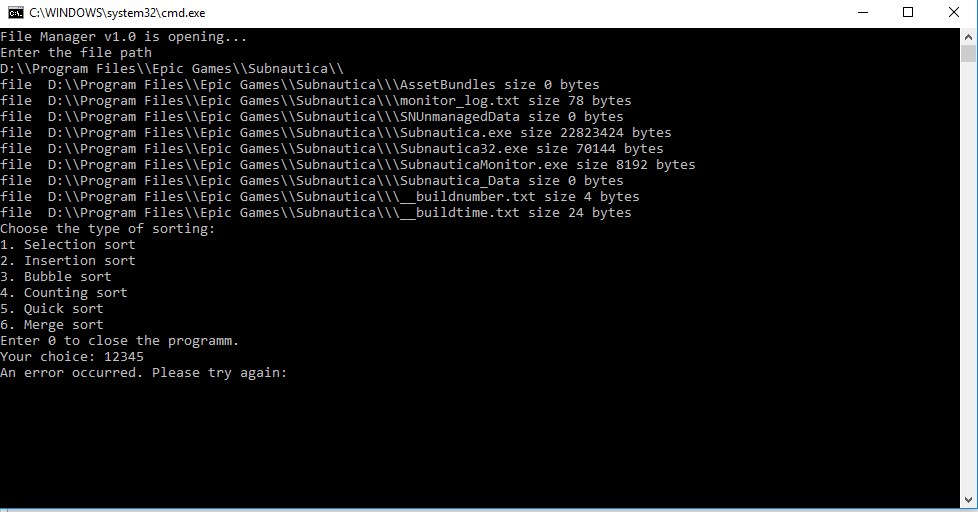


1. Ошибка при вводе неправильного пути.

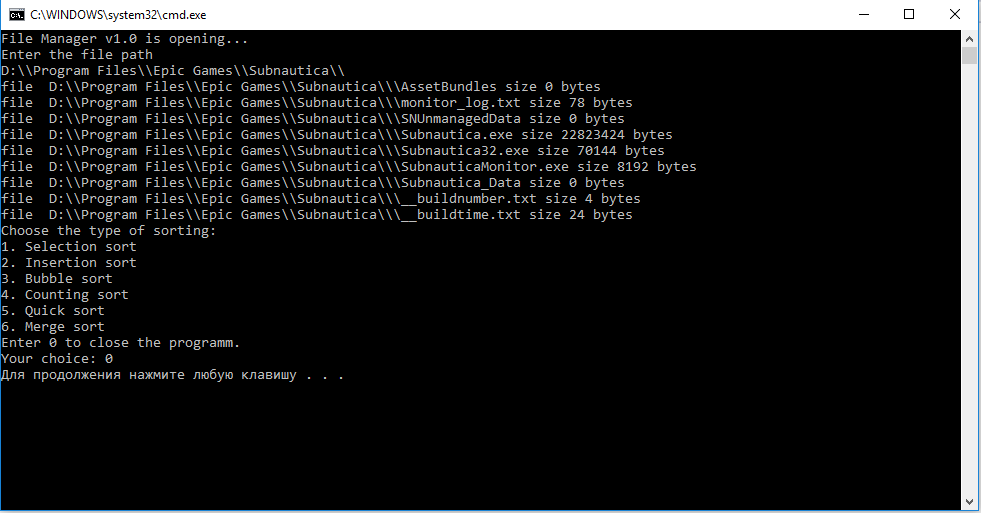
Программа выведет файлы из указанной папки с размерами в байтах и меню, включающее в себя все методы сортировки, используемые в программе (Рис.3). Далее вам следует выбрать номер сортировки в списке и нажать Enter. Если выбран неверный номер, то программа попросит заново, сообщив об ошибке (Рис.4).Если ввести 0, то программа завершит свою работу (Рис.5).



1. Вывод списка файлов и меню.

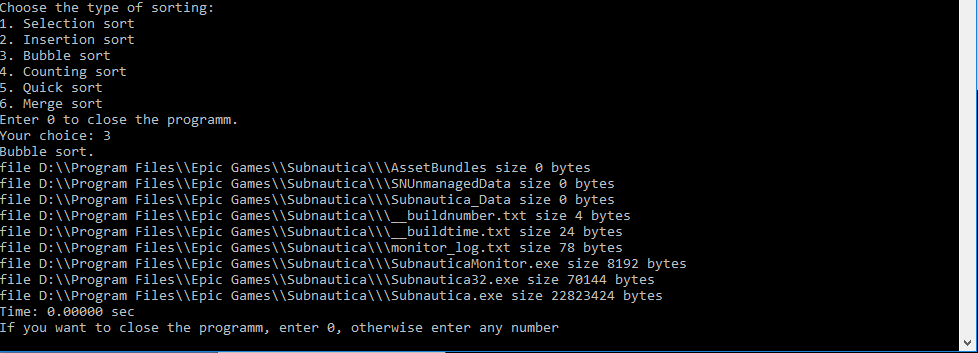


1. Ошибка в выборе сортировки.



1. Выход из программы.

Когда вы выберете метод сортировки, программа выведет отсортированный список файлов по возрастанию их размеров из этой папки, а также время сортировки (Рис.6). Если вы хотите выйти из программы, введите 0 и нажмите Enter, в противном случае, введите любое число, кроме нуля и нажмите Enter, тогда программа опять попросит путь к папке на вашем компьютере (см. пункт 1).



1. Вывод отсортированного списка и времени сортировки.

# Руководство программиста.

## Структура программы.

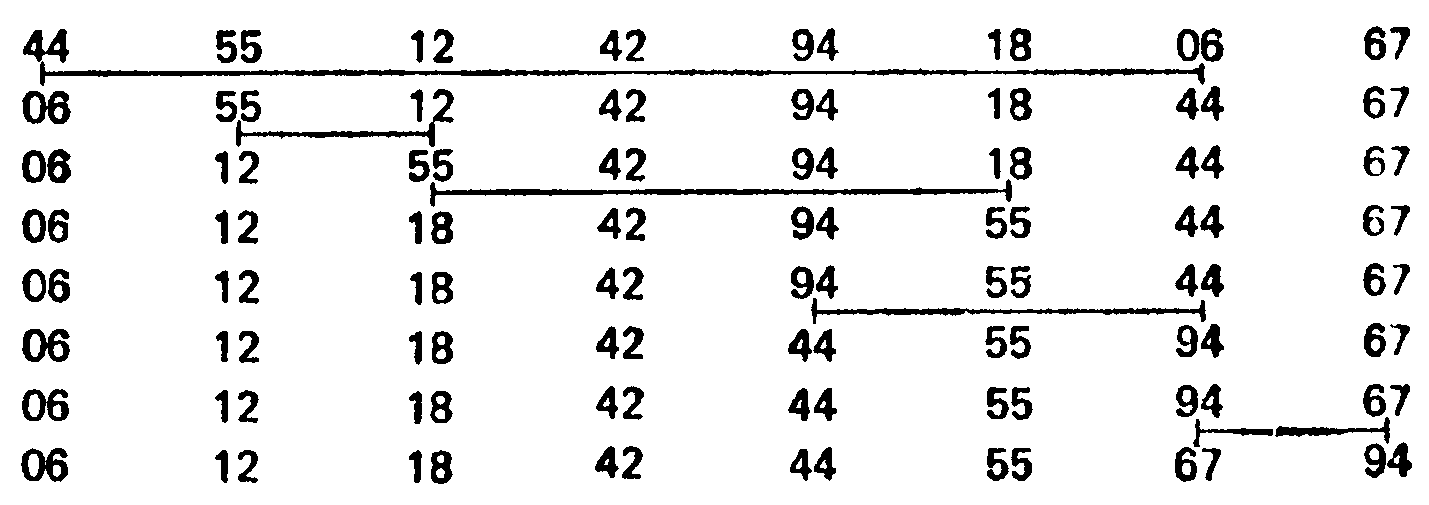
Файл **main.c**, в котором содержится весь код программы.

## Описание алгоритмов.

В этом пункте содержится описание всех методов сортировки, используемых в программе.

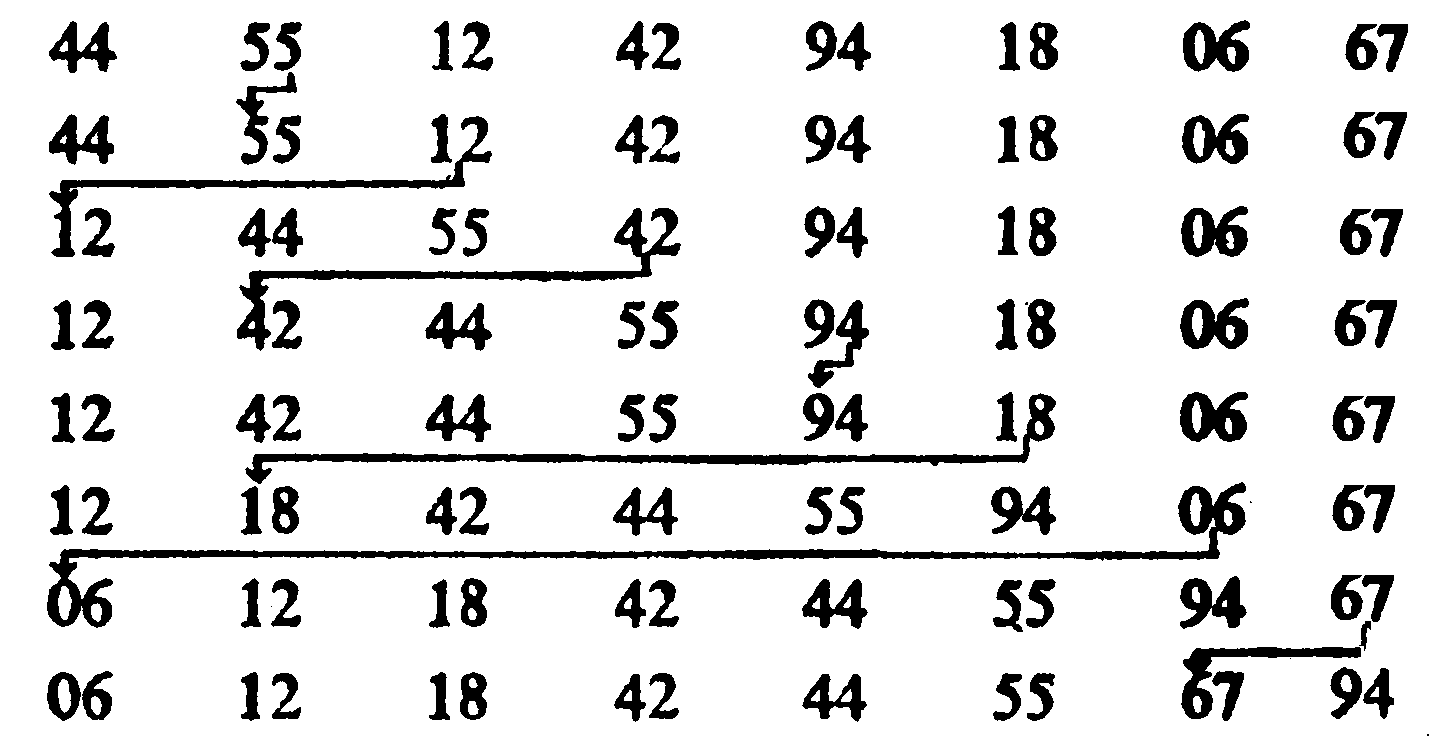
### Сортировка выбором.

Сортировка выбором основана на линейном поиске минимального элемента в массиве, после которого этот элемент меняется с крайним левым элементом массива. После чего левая часть, состоящая из одного элемента, считается отсортированной. Затем эта часть мысленно отбрасывается, и те же действия проводятся с остальными элементами массива. Пример показан на рисунке:



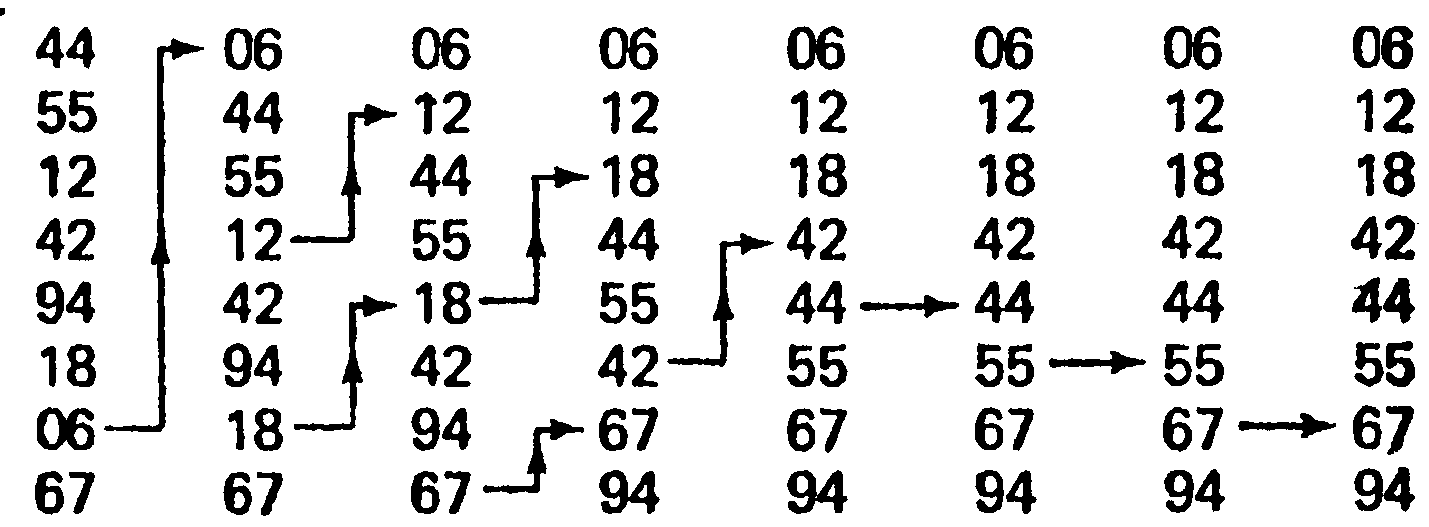
### Сортировка простыми вставками.

Сортировка простыми вставками основана на принципе: крайний левый элемент массива считается упорядоченным массивом из одного элемента, затем берется следующий элемент и «прогоняется» по упорядоченной части (если меньше, то меняется местами с предыдущим). Так происходит до тех пор, пока слева от этого элемента не окажется большее число или конец массива. Такие операции проводятся с каждым элементом массива и упорядоченной частью массива, стоящей перед ним. Пример показан на рисунке:



### Сортировка пузырьком.

Алгоритм данной сортировки основан на «всплытии» меньшего элемента массива. Массив представляется вертикально, затем нижний элемент сравнивается с элементом выше. Если выбранный элемент меньше, то они меняются местами. Так продолжается до тех пор, пока элемент не встретит меньший элемент. Далее все те же операции проводятся для элементов ниже «всплывшего» с самого начала. Пример на рисунке:



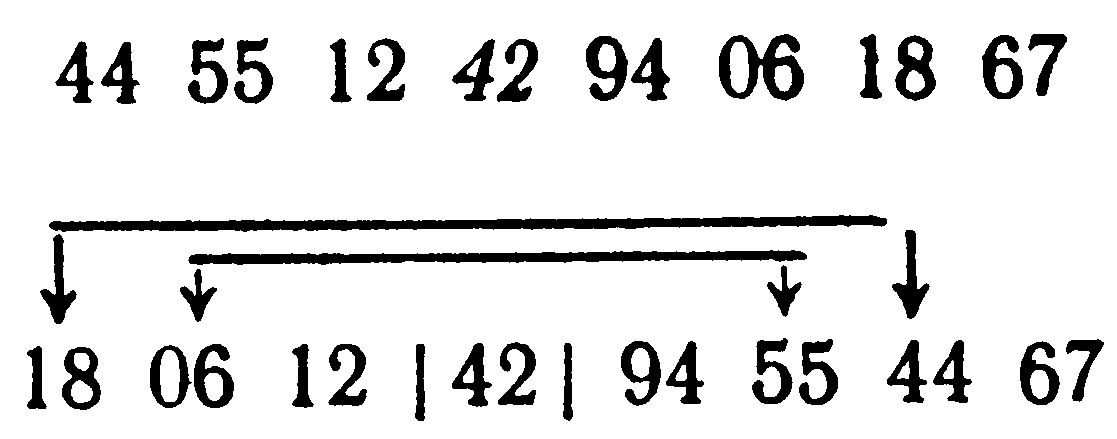
### Сортировка подсчетом.

Алгоритм этой сортировки основан на том, что известен диапазон значений элементов массива. Делается проход по массиву и подсчитывается количество одинаковых элементов, которое записывается в новый массив (по его индексам можно узнать, какой элемент там подсчитан). Затем делается проход по новому массиву, и в исходный записывается столько элементов, сколько написано в массиве подсчета. Пример в таблице (диапазон значений от 0 до 8):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Индексы | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 шаг | Исходный массив: | | | | | | | | |
|  | 4 | 6 | 1 | 0 | 7 | 5 | 3 | 1 | 2 |
| 2 шаг | Массив подсчета (элемент 0 индекса – кол-во нулей и т.д.): | | | | | | | | |
|  | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 шаг | Отсортированный массив: | | | | | | | | |
|  | 0 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

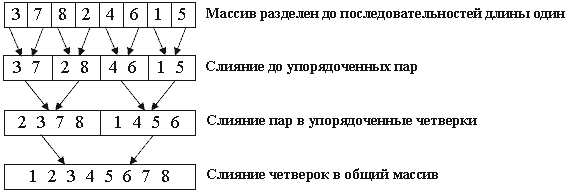
### Быстрая сортировка.

Алгоритм быстрой сортировки основан на выборе элемента массива (случайно или центрального), а затем перемещении всех элементов, меньших его, влево, а всех элементов, больших его, вправо от самого элемента. Затем рассматривается каждая такая часть (в одной из них сам выбранный элемент), и для нее производятся абсолютно те же операции, пока часть массива не станет состоять из одного элемента (такая часть считается отсортированной). Пример одного действия на рисунке (остальные аналогично):



### Сортировка слиянием.

Исходный массив разбивается на 2 массива пополам нацело, далее – каждая часть еще на 2 массива. Так продолжается до тех пор, пока части не будут состоять из одного элемента. Затем две такие части «сливают» в массив (упорядоченно). Так делают со всеми частями, т.е. проходят обратно, в итоге получая исходный отсортированный массив. Пример:



## Описание фукций

void choose(ULONGLONG\* filesize, int kolvo, int\* newindex)

Назначение: сортировка выбором для указанного массива указанного размера.

Входные параметры: filesize – массив размеров файлов, которые нужно упорядочить; kolvo – количество упорядочиваемых файлов; newindex – массив индексов файлов.

Выходные параметры: указатель на массив индексов файлов.

void insert(ULONGLONG\* filesize, int\* newindex, int kolvo)

Назначение: сортировка простыми ставками для массива указанного размера.

Входные параметры: filesize – массив размеров файлов, которые нужно упорядочить; kolvo – количество упорядочиваемых файлов; newindex – массив индексов файлов.

Выходные параметры: указатель на массив индексов файлов.

void bubblesort(ULONGLONG\* filesize, int\* newindex, int kolvo)

Назначение: сортировка пузырьком для массива указанного размера.

Входные параметры: filesize – массив размеров файлов, которые нужно упорядочить; kolvo – количество упорядочиваемых файлов; newindex – массив индексов файлов.

Выходные параметры: указатель на массив индексов файлов.

void countingsort(ULONGLONG\* a, int kolvo, int\* newindex)

Назначение: сортировка подсчетом для массива указанного размера.

Входные параметры: filesize – массив размеров файлов, которые нужно упорядочить; kolvo – количество упорядочиваемых файлов; newindex – массив индексов файлов.

Выходные параметры: указатель на массив индексов файлов.

int\* quicksort(ULONGLONG\* size, int n1, int n2, int n, int\* id)

Назначение: быстрая сортировка для массива указанного размера.

Входные параметры: filesize – массив размеров файлов, которые нужно упорядочить; n – количество упорядочиваемых файлов; newindex – массив индексов файлов; n1 – первый индекс; n2 – последний индекс.

Выходные параметры: указатель на массив индексов файлов.

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

Назначение: вспомогательная функция для сортировки слиянием. Соединяет подмассивы.

Входные параметры: a - массив индексов; size - массив размеров файлов; l – индекс первого элемента; m – индекс среднего элемента; r – индекс последнего элемента.

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

Назначение: сортировка слиянием для массива.

Входные параметры: a - массив индексов; size - массив размеров файлов; l – индекс первого элемента; r – индекс последнего элемента.

Выходные параметры: указатель на массив индексов файлов.

void zeroing(ULONGLONG\* filesize, ULONGLONG\* filesizecopy, int\* filesindex, int\* newindex, int kolvo)

Назначение: обнуление массивов индексов и размеров файлов.

Входные параметры: количество файлов и массивы их исходных размеров и наименований

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

Назначение: подсчет количества элементов в указанной папке и получение их названий и размеров.

Входные параметры: sDir – строка, содержащая путь к папке; fileNames, filesizes – массивы для записи полных имен и размеров файлов.

void print(ULONGLONG\* filesize, wchar\_t\*\* fileNames, int kolvo)

Назначение: печать названий файлов.

Входные параметры: размеры файлов, количество файлов, названия файлов.

void menu()

Назначение: вывод меню.

Входные параметры: -

Выходные параметры: -

# Заключение.

Была разработана программа-прототип файлового менеджера для сортировки файлов в указанной папке по их размеру. Использовались следующие методы сортировки (в скобках указано время их работы):

1. Сортировка выбором (O()).
2. Сортировка простыми вставками (O()).)).
3. Сортировка пузырьком (O()).
4. Сортировка подсчетом (O(n)).
5. Быстрая сортировка (O(n \* log(n))).
6. Сортировка слиянием (O(n \* log(n))).