## 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

Отчёт по лабораторной работе

Методы сортировки

Выполнил:

студент гр. 381806-1

Бакаева М.И.

Проверил:

Доцент каф. МОСТ, ИИТММ

Кустикова В.Д.

Нижний Новгород

2018

Содержание

[Введение 3](#_Toc533376265)

[1. Постановка задачи 4](#_Toc533376266)

[2. Руководство пользователя 5](#_Toc533376267)

[3. Руководство программиста 7](#_Toc533376268)

[3.1. Структура программы 7](#_Toc533376269)

[3.2. Описание алгоритма 7](#_Toc533376270)

[3.2.1. Сортировка выбором 7](#_Toc533376271)

[3.2.2. Сортировка вставками 8](#_Toc533376272)

[3.2.3. Сортировка пузырьком 9](#_Toc533376273)

[3.2.4. Сортировка подсчетом 10](#_Toc533376274)

[3.2.5. Быстрая сортировка 11](#_Toc533376275)

[3.2.6. Сортировка слиянием 12](#_Toc533376276)

[3.3. Описание функций 13](#_Toc533376277)

[Заключение 15](#_Toc533376278)

[Литература 16](#_Toc533376279)

# Введение

Вся человеческая деятельность связана с поиском и упорядочиванием. Почему? Всё просто, работать с упорядоченным массивом значительно эффективнее. Зачастую успех деятельности зависит от быстроты выбора правильного решения. Поэтому, если у вас имеется упорядоченная информация, вы скорее всего достигнете больших успехов.

Применение методов сортировок удобнее всего рассмотреть на простом файловом менеджере, который сможет упорядочивать файлы по их размеру. После этого вы легко сможете ориентировать по заданному каталогу.

# Постановка задачи

**Задача:** Разработать файловый менеджер с функцией показа файлов в заданном каталоге упорядоченным по возрастанию или убыванию размера.

**Входные данные:**

1. Путь до директории, где необходимо отсортировать содержимое
2. Метод сортировки

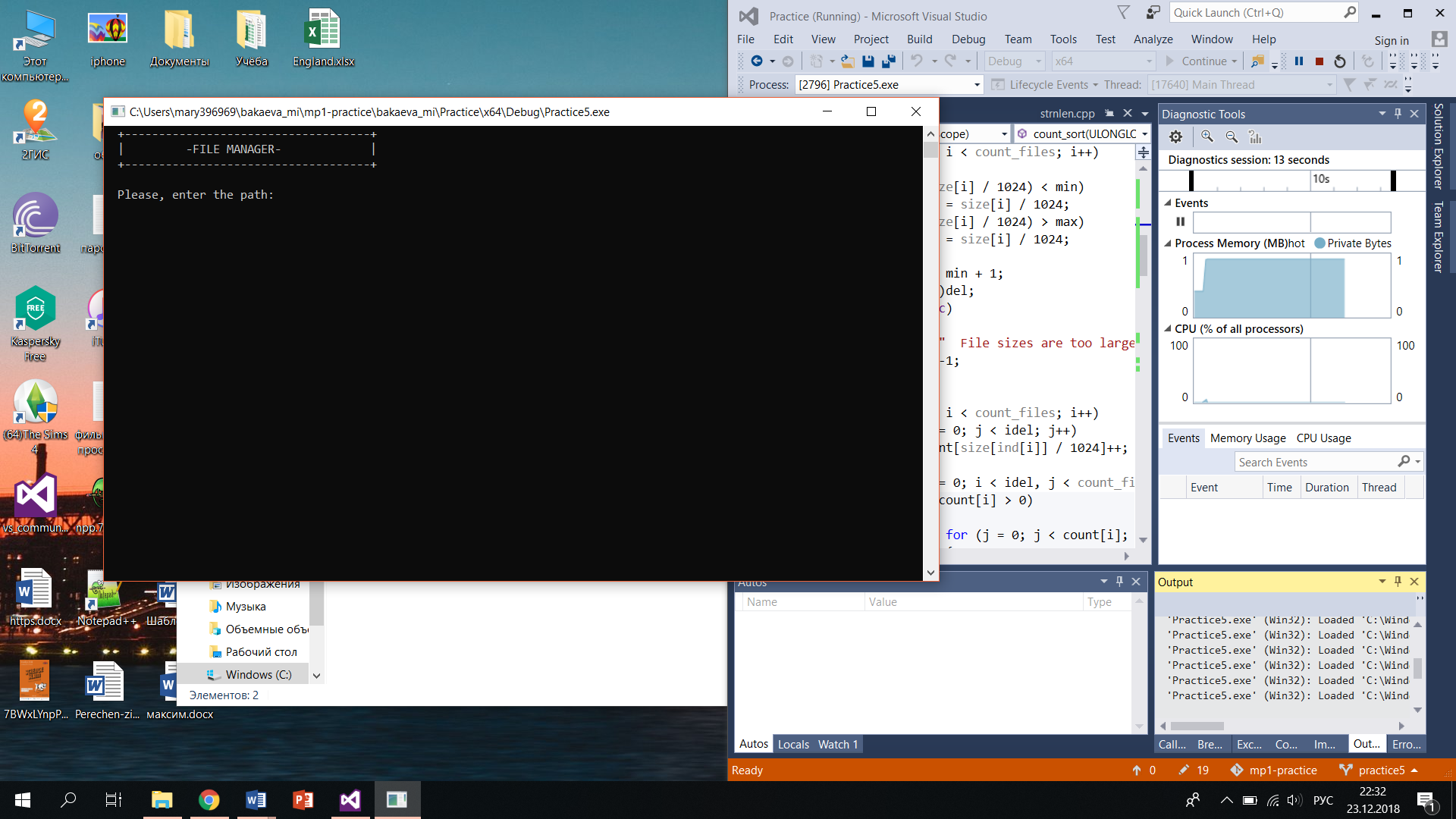
**Выход:**

1. Отсортированный список имен файлов с указанием размера
2. Время сортировки

# Руководство пользователя

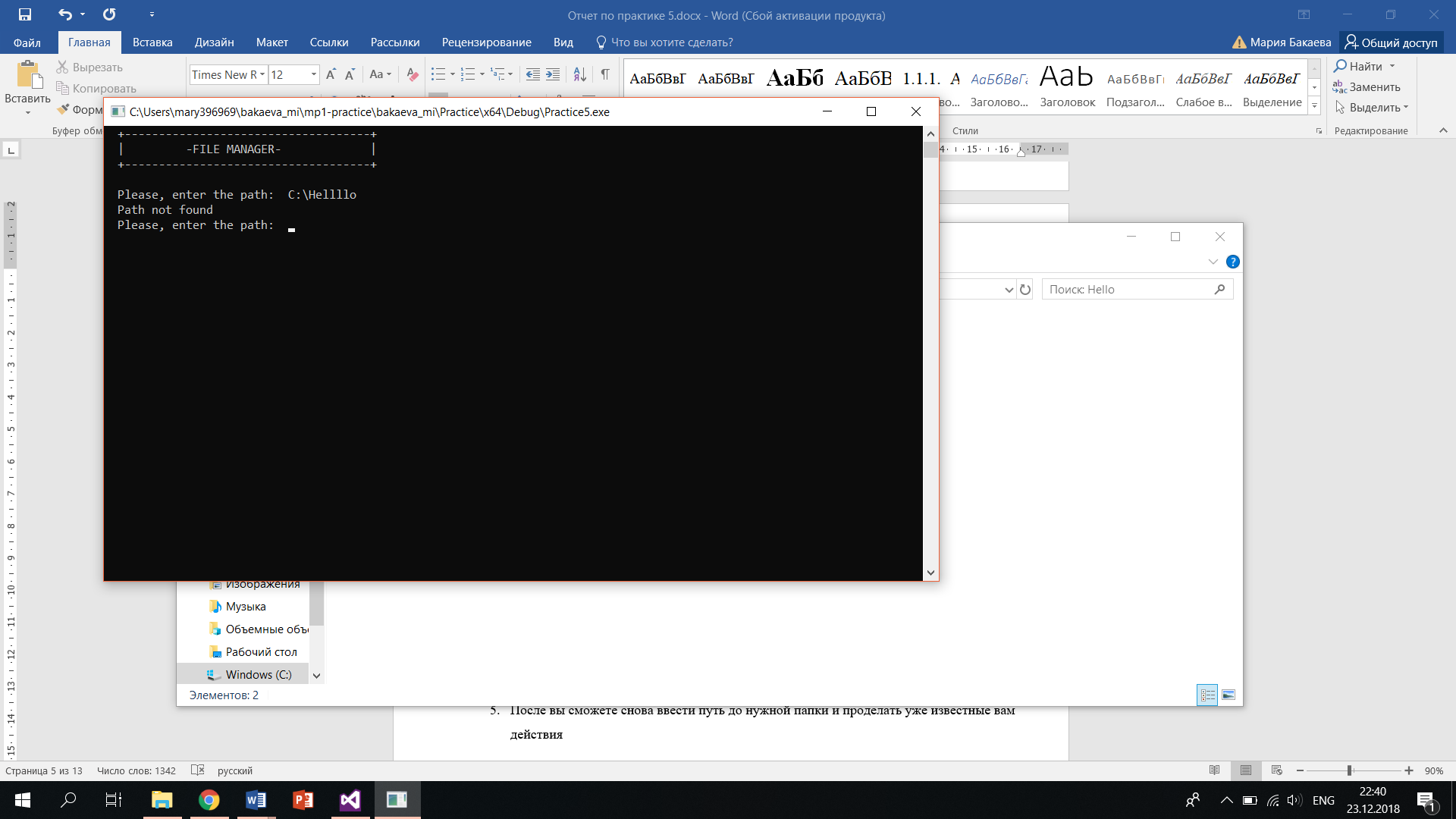
Далее будет представлена пошаговая инструкция по работе с программой

1. Запустите приложение Practice5.exe, при запуске вы увидите название самой программы и просьбу ввести путь до нужной папки (Рис 1)

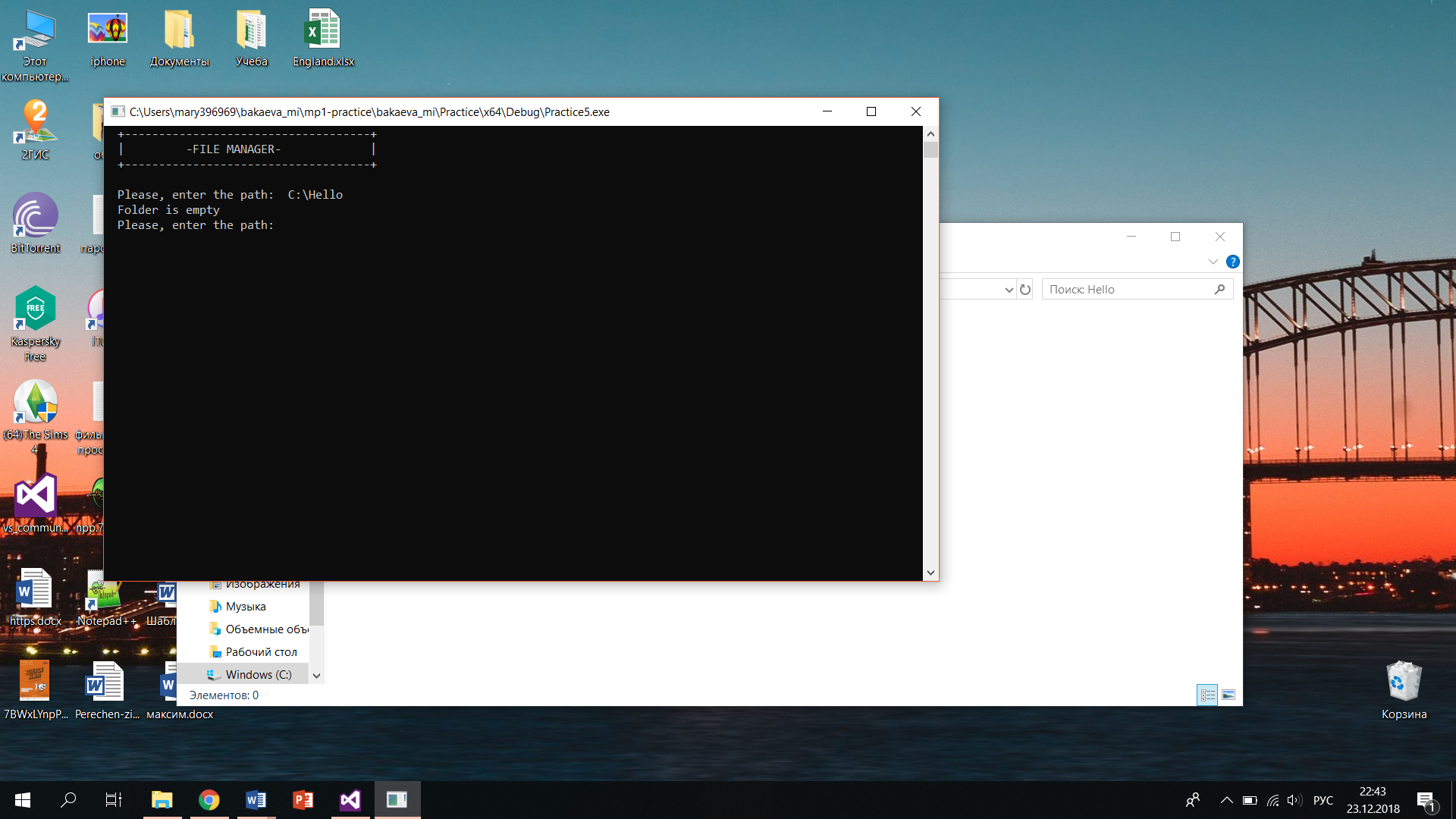


* 1. Программа после запуска

1. Если введенный путь не существует, программа сообщит вам об этом (Рис 2). Попробуйте ввести путь еще раз, проверьте его на правильность ввода. Если в указанная папка будет пуста, выведется соответствующее сообщение (Рис 3).

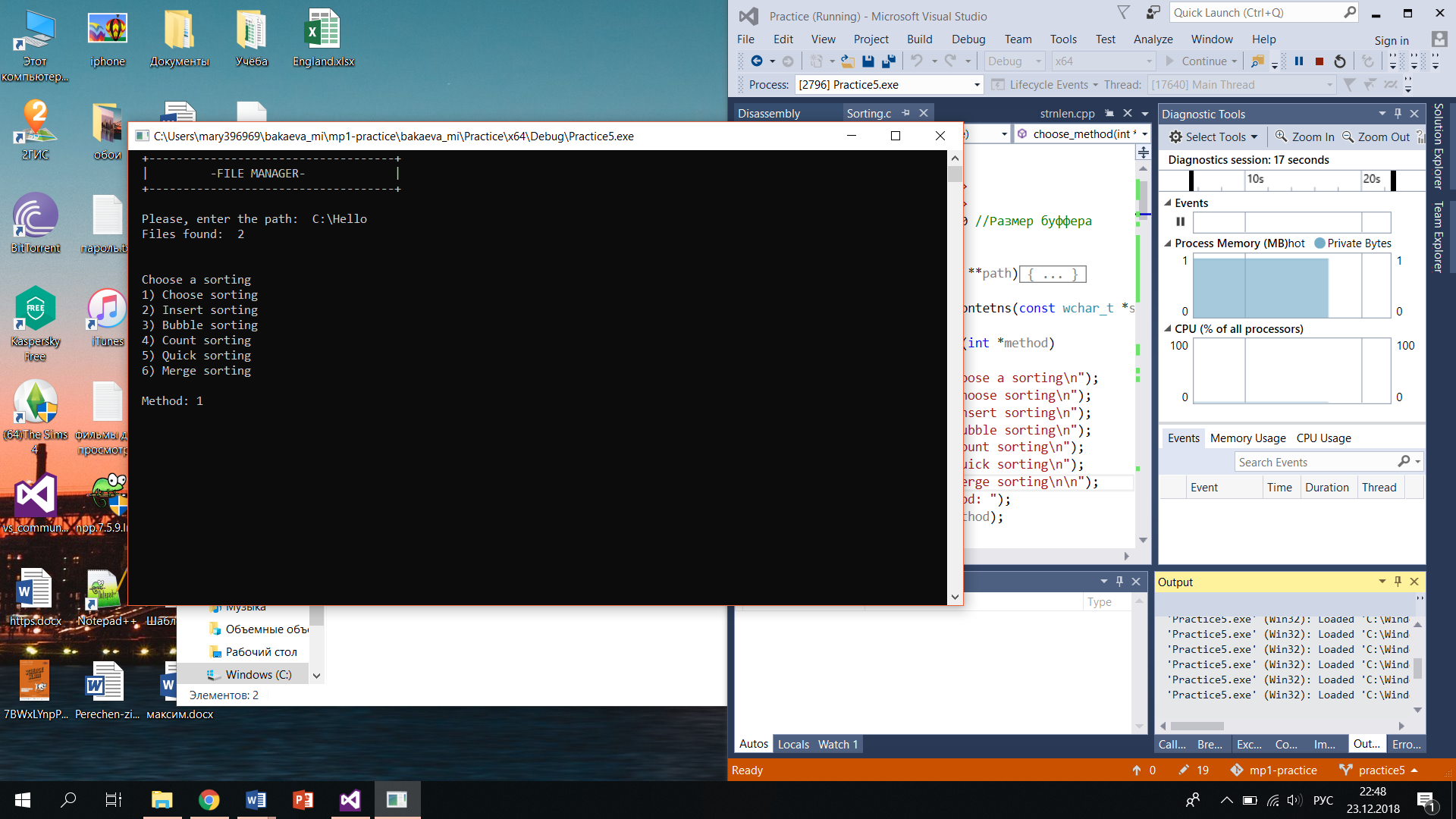


* 1. Несуществующий путь



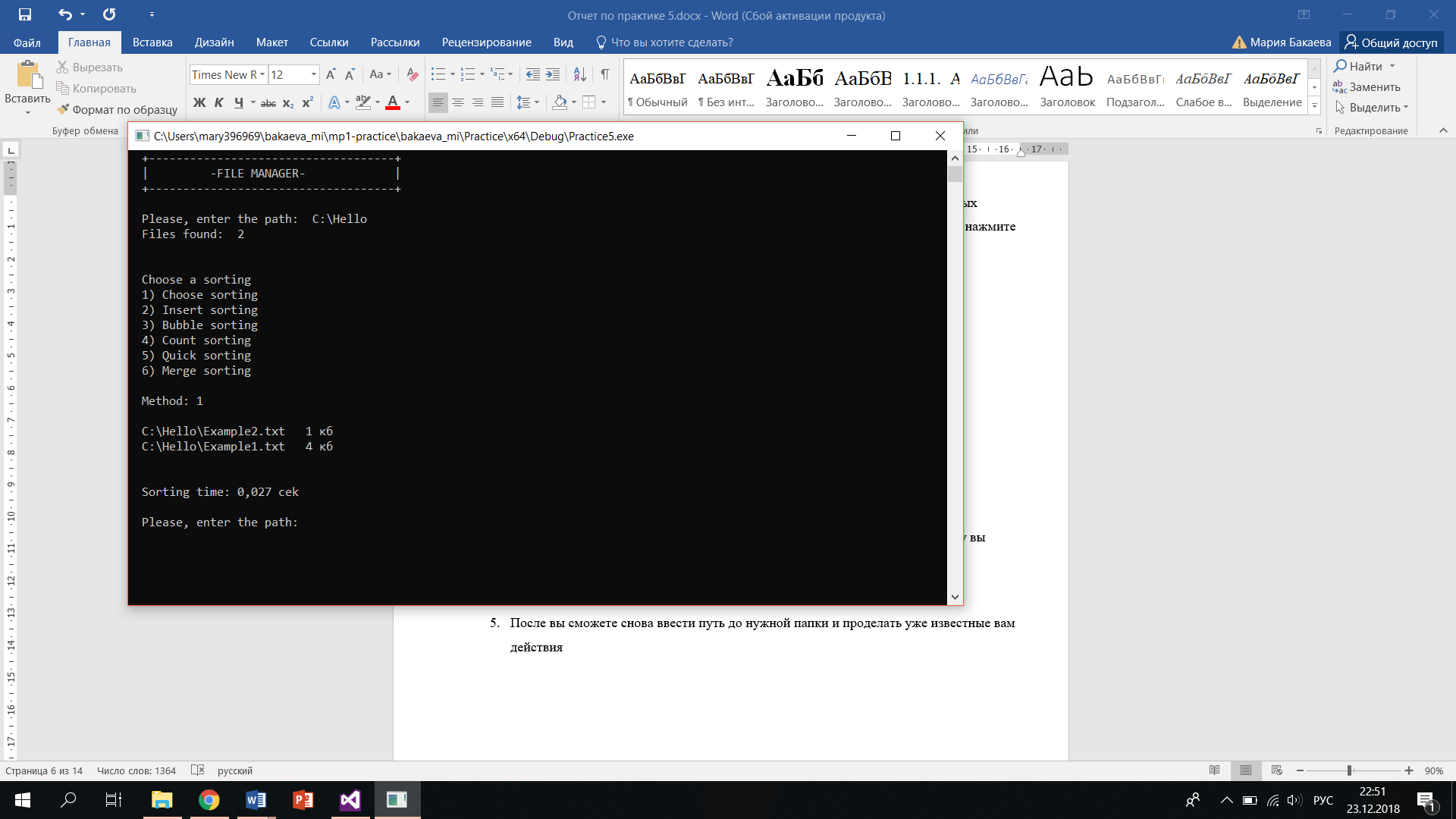
* 1. Пустая папка

1. После корректного ввода пути выберите способ сортировки из 6 предложенных (Рис 4). Для этого наберите цифру, соответствующую выбранному способу и нажмите Enter



* 1. Список сортировок после ввода непустой папки

1. На экране появятся отсортированные по возрастанию размера файлы (Рис 5). Снизу вы сможете увидеть время, которое было затрачено на сортировку. После вы сможете снова ввести путь до нужной папки и проделать уже известные вам действия



* 1. Отсортированные файлы и запрос на новый путь

# Руководство программиста

### Структура программы

Весь код программы находится в файле **Sorting.c**

### Описание алгоритма

### Сортировка выбором

Идея метода состоит в том, чтобы создавать отсортированную последовательность путем присоединения элементов в правильном порядке.

На i-том шаге выбирается наименьший элемент из a[i] до a[n]. Он меняется местами с a[i]. На n – 1 шаге вся последовательность, кроме a[n] уже отсортирована. А a[n] уже стоит на своем последнем месте, так как все меньшие элементы ушли влево.

Пример сортировки выборов приведен в Таблица 1.

1. Пример сортировки выбором

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность, n = 6** | | | | | | **min** | **min ‹—› a[i]** |
| Исходная последовательность | 4 | 8 | 7 | 1 | 4 | 3 |  |  |
| 0 | **1** | 8 | 7 | 4 | 4 | 3 | 1 | 4 |
| 1 | **1** | **3** | 7 | 4 | 4 | 8 | 3 | 8 |
| 2 | **1** | **3** | **4** | 7 | 4 | 8 | 4 | 7 |
| 3 | **1** | **3** | **4** | **4** | 7 | 8 | 7 | 7 |
| 4 | **1** | **3** | **4** | **4** | **7** | 8 | 8 | 8 |
| 5 | **1** | **3** | **4** | **4** | **7** | **8** | - | - |

### Сортировка вставками

Будем считать, что первый элемент последовательности отсортирован, а оставшаяся часть последовательности нет. Сравниваем a[i] элемент с a[i + 1], и если a[i + 1] меньше отсортированного, то меняем их местами. Если же a[i + 1] больше a[i], то считаем, что он теперь тоже отсортирован. Проделываем те же действия с оставшимися элементами. Пример сортировки вставками приведен в Таблица 2.

1. Пример сортировки вставками

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность, n = 6** | | | | | | **Сравниваются** | **Замена места** |
| Исходная последовательность | 4 | 8 | 7 | 1 | 4 | 3 |  |  |
| 0 | **4** | 8 | 7 | 1 | 4 | 3 | 4 и 8 | нет |
| 1 | **4** | **8** | 7 | 1 | 4 | 3 | 8 и 7 | да |
| **4** | 7 | **8** | 1 | 4 | 3 | 7 и 4 | нет |
| 2 | **4** | **7** | **8** | 1 | 4 | 3 | 8 и 1 | да |
| **4** | **7** | 1 | **8** | 4 | 3 | 1 и 7 | да |
| **4** | 1 | **7** | **8** | 4 | 3 | 1 и 4 | да |
| 3 | **1** | **4** | **7** | **8** | 4 | 3 | 8 и 4 | да |
| **1** | **4** | **7** | 4 | **8** | 3 | 4 и 7 | да |
| **1** | **4** | 4 | **7** | **8** | 3 | 4 и 4 | нет |
| 4 | **1** | **4** | **4** | **7** | **8** | 3 | 8 и 3 | да |
| **1** | **4** | **4** | **7** | 3 | **8** | 3 и 7 | да |
| **1** | **4** | **4** | 3 | **7** | **8** | 3 и 4 | да |
| **1** | **4** | 3 | **4** | **7** | **8** | 3 и 4 | да |
| **1** | 3 | **4** | **4** | **7** | **8** | 3 и 1 | нет |
| **1** | **3** | **4** | **4** | **7** | **8** | - | - |

### Сортировка пузырьком

Расположим массив сверху вниз, от нулевого элемента к последнему. Идем по последовательности снизу - вверх, просматривая при том соседние пары элементов. Если элементы пары находятся в неправильном порядке, меняем их местами. После нулевого прохода вверху оказывается самый маленький элемент. Следующий проход делается до второго сверху элемента. Проходы совершаются до тех пор, пока в нем не останется только 1 элемент. Пример сортировки пузырьком приведен в Таблица 3.

1. Пример сортировки пузырьком

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность, n = 6** | | | | | | **Сравниваемая пара** | **Замена места** |
| Исходная последовательность | 4 | 8 | 7 | 1 | 4 | 3 |  |  |
| 0 | 4 | 8 | 7 | 1 | **4** | **3** | 3 и 4 | да |
| 4 | 8 | 7 | **1** | **3** | 4 | 3 и 1 | нет |
| 4 | 8 | **7** | **1** | 3 | 4 | 1 и 7 | да |
| 4 | **8** | **1** | 7 | 3 | 4 | 1 и 8 | да |
| **4** | **1** | 8 | 7 | 3 | 4 | 1 и 4 | да |
| 1 | 1 | 4 | 8 | 7 | **3** | **4** | 4 и 3 | нет |
| 1 | 4 | 8 | **7** | **3** | 4 | 3 и 7 | да |
| 1 | 4 | **8** | **3** | 7 | 4 | 3 и 8 | да |
| 1 | **4** | **3** | 8 | 7 | 4 | 3 и 4 | да |
| 2 | 1 | 3 | 4 | 8 | **7** | **4** | 4 и 7 | да |
| 1 | 3 | 4 | **8** | **4** | 7 | 4 и 8 | да |
| 1 | 3 | **4** | **4** | 8 | 7 | 4 и 4 | нет |
| 3 | 1 | 3 | 4 | 4 | **8** | **7** | 7 и 8 | да |
| 1 | 3 | 4 | **4** | **7** | 8 | 7 и 4 | нет |
| 5 | 1 | 3 | 4 | 4 | 7 | 8 | - | - |

### Сортировка подсчетом

Находим min и max в сортируемой последовательности. Создаем дополнительный массив (пусть он называется С) размером (max – min) + 1 и заполняем его нулями. Последовательно пройдем по заданному массиву и запишем в С[i] количество чисел равных i. Теперь достаточно пройти по массиву С и для каждого i в заданный массив записать последовательно число С[i] раз. Пример сортировки подсчетом приведен в Таблица 4.

1. Пример сортировки подсчетом

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность, n  = 6, min = 1, max = 8** | | | | | | **Дополнительный массив** | | | | |
| Исходная последовательность | 4 | 0 | 3 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 4 |
| 1 | + |  |  |  |  |  | 0 | 0 | 0 | 0 | 1 |
|  | + |  |  |  |  | 1 | 0 | 0 | 0 | 1 |
|  |  | + |  |  |  | 1 | 0 | 0 | 1 | 1 |
|  |  |  | + |  |  | 1 | 1 | 0 | 1 | 1 |
|  |  |  |  | + |  | 1 | 1 | 1 | 1 | 1 |
|  |  |  |  |  | + | 1 | 1 | 1 | 2 | 1 |
| 2 | **0** | 0 | 3 | 1 | 2 | 3 | + |  |  |  |  |
| **0** | **1** | 3 | 1 | 2 | 3 |  | + |  |  |  |
| **0** | **1** | **2** | 1 | 2 | 3 |  |  | + |  |  |
| **0** | **1** | **2** | **3** | **3** | 3 |  |  |  | + |  |
| **0** | **1** | **2** | **3** | **3** | **4** |  |  |  |  | + |
|  | **0** | **1** | **2** | **3** | **3** | **4** |  |  |  |  |  |

### Быстрая сортировка

1. Введем два указателя i и j. В начале сортировки они указывают на левый и правый концы последовательности соответственно. Также выберем опорный элемент p, по которому будет производиться разделение (p может быть серединой последовательности или любым другим элементом).
2. Будем двигать указатель i на 1 шаг по направлению к концу, пока не будет найден элемент a[i] ≥ p. Затем аналогично начинаем двигать указать j по направлению к левому концу, пока не будет найден a[j] ≤ p.
3. Далее, если i ≤ j, меняем местами a[i] и a[j]. Далее продолжаем двигать i и j по тому же алгоритму.
4. Повторяем шаг 3, пока i ≤ j.
5. Далее выполняем рекурсивную сортировку каждой части массива

В Таблица 5 приведен пример быстрой сортировки.

1. Пример быстрой сортировки

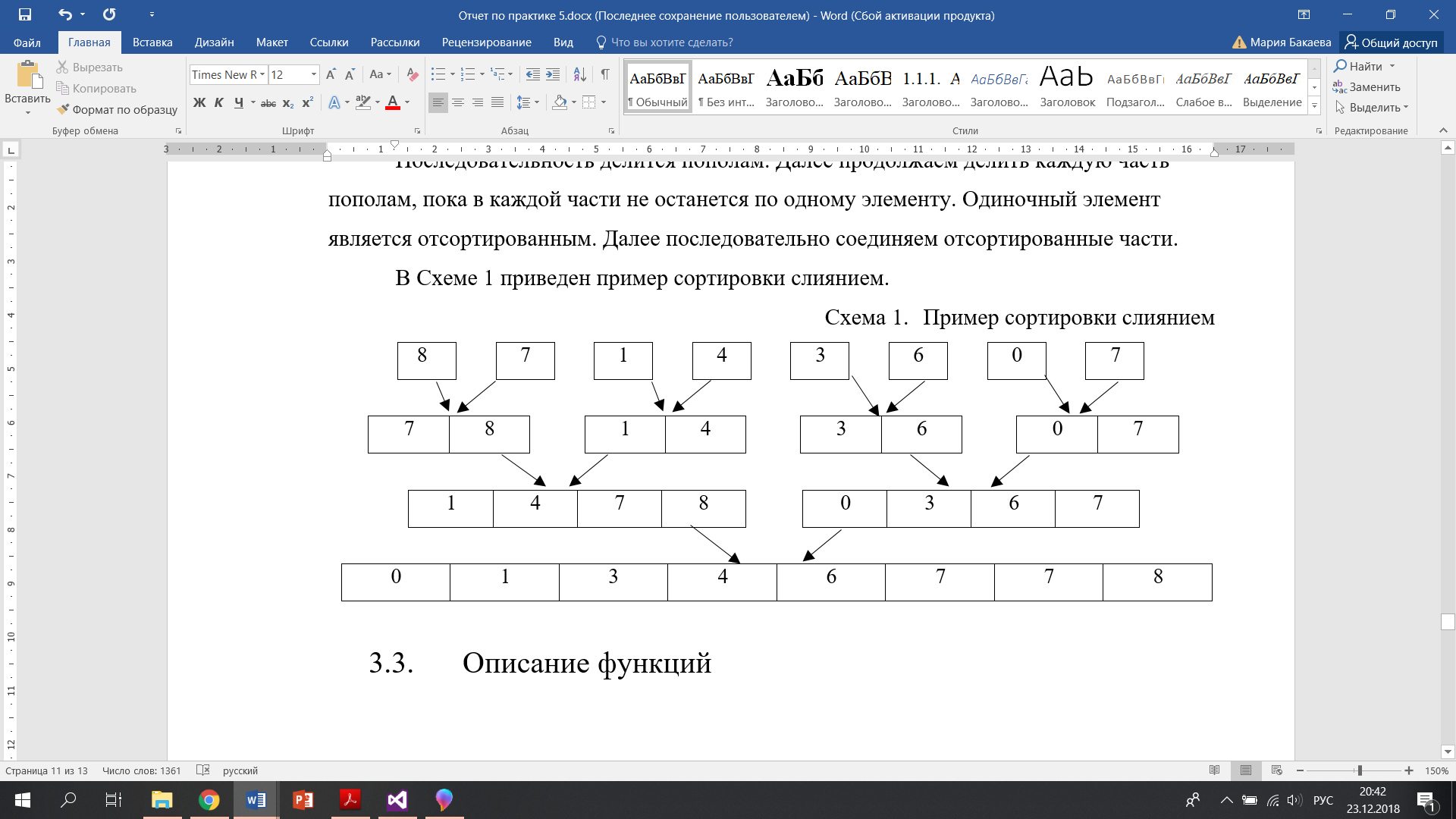
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** |  | | **Последовательность, n = 6** | | | | | |
| Исходная последовательность | 4 | 8 | | 7 | 6 | 2 | 9 | 3 |
| 0 | 4 | 8 | | 7 | **6** | 2 | 9 | 3 |
| 4 | 3 | | 7 | **6** | 2 | 9 | 8 |
| 4 | 3 | | 2 | **6** | 7 | 9 | 8 |
| 1 | 4 | **3** | | 2 | 6 | 7 | **9** | 8 |
| 2 | **3** | | 4 | 6 | 7 | 8 | **9** |
|  | 2 | 3 | | 4 | 6 | 7 | 8 | 9 |

### Сортировка слиянием

Последовательность делится пополам. Далее продолжаем делить каждую часть пополам, пока в каждой части не останется по одному элементу. Одиночный элемент является отсортированным. Далее последовательно соединяем отсортированные части.

В Схема 1приведен пример сортировки слиянием (от исходного множества к отсортированному).

1. Пример сортировки слиянием



## Описание функций

void input(wchar\_t \*\*path)

**Назначение:** Запрос у пользователя пути к нужной папке для сортировки

**Входные данные:** \*\*path – указатель на массив, в который будет записана строка с директорией

int ListDirectoryContetns(const wchar\_t \*sDir, wchar\_t \*\*\*fNames, ULONGLONG \*\*fSizes)

**Назначение:** Получение имен и размеров файлов, подсчет количества файлов

**Входные данные:** \*sDir – строка с полученной директорией, \*\*\*fNames – массив для записи имён файлов, \*\*fSizes – массив для записи размеров файлов

**Выходные данные:** Количество файлов в папке, число -1 при несуществующей директории

void choose\_method(int \*method)

**Назначение:** Вывод списка сортировок на экран, получение через указатель выбранного метода

**Входные данные:** \*method – указатель на переменную, содержащую номер выбранной сортировки

void output(int \*ind, int count\_files, wchar\_t \*\*fNames, ULONGLONG \*size)

**Назначение:** Вывод отсортированного списка файлов с размерами

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*\*fNames – массив с именами файлов, \*size – массив с размерами файлов

void choose\_sort(int count\_files, ULONGLONG \*size, int \*ind)

**Назначение:** Выполнение сортировки выбором

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*size – массив с размерами файлов

void insert\_sort(ULONGLONG \*size, int count\_files, int \*ind)

**Назначение:** Выполнение сортировки вставками

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*size – массив с размерами файлов

void bubble\_sort(ULONGLONG \*size, int count\_files, int \*ind)

**Назначение:** Выполнение сортировки «пузырьком»

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*size – массив с размерами файлов

int count\_sort(ULONGLONG \*size, int count\_files, int \*ind)

**Назначение:** Выполнение сортировки подсчетом

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*size – массив с размерами файлов

void quick\_split(int \*ind, ULONGLONG \*size, int \*i, int \*j, ULONGLONG p)

**Назначение:** Подготовка к быстрой сортировки

**Входные данные:** \*ind - массив для индексов отсортированных файлов, \*size – массив с размерами файлов, i и j – индексы крайних элементов сортируемого фрагмента, p – опорный элемент, по которому будет производиться разделение массива

void quick\_sort(ULONGLONG \*size, int \*ind, int n1, int n2)

**Назначение:** Выполнение быстрой сортировки

**Входные данные:** \*ind - массив для индексов отсортированных файлов, \*size – массив с размерами файлов, n1 и n2 – индексы крайних элементов сортируемого фрагмента

void merge(ULONGLONG \*size, int lb, int mid, int ub, int \*ind)

**Назначение:**

**Входные данные:** \*ind - массив для индексов отсортированных файлов, \*size – массив с размерами файлов, lb и ub – индексы крайних элементов сортируемого фрагмента, mid – индекс среднего элемента сортируемого фрагмента

void merge\_sort(ULONGLONG \*size, int lb, int ub, int \*ind)

**Назначение:** Выполнение сортировки слиянием

* **Входные данные:** \*ind - массив для индексов отсортированных файлов, \*size – массив с размерами файлов, lb и ub – индексы крайних элементов сортируемого фрагмента

void main()

**Назначение:** Основная функция

**Входные данные: -**

# Заключение

В ходе практической работы я разработала и реализовала прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченном по возрастанию размера. В ходе работы были использовано 6 сортировок:

1. Сортировка выбором
2. Сортировка вставками
3. Сортировка «пузырьком»
4. Сортировка подсчетом
5. Быстрая сортировка
6. Сортировка слиянием

Далее я проанализировала представленные выше методы сортировки (Таблица 6)

1. Сложность сортировок

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Сортировка пузырьком** | **Сортировка выбором** | **Сортировка вставками** | **Быстрая сортировка** | **Сортировка слиянием** | **Сортировка подсчетом** |
| **Время выполнения** | O(N2) | O(N2) | O(N2) | O(Nlog N) | O(Nlog N) | O(N) |
| **Дополнительная память** | O(1) | O(1) | O(1) | O(log N) | O(N) | O(N) |

На графиках приведена зависимость времени работы сортировки от количества сортируемых элементов

# Литература

1. http://algolist.manual.ru/