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: Параничева Алена Владиславовна

студент группы 381806-1

Проверил: к. т. н., доцент кафедры МОСТ

Кустикова Валентина Дмитриевна

Нижний Новгород

2018

**Содержание:**

Оглавление

[Введение: 3](#_Toc533625478)

[2. Постановка задачи: 4](#_Toc533625479)

[3. Руководство пользователя 5](#_Toc533625480)

[4. Руководство программиста 7](#_Toc533625481)

[4.1. Структура программы 7](#_Toc533625482)

[4.2. Описание алгоритма 7](#_Toc533625483)

[4.2.1. Сортировка выбором 7](#_Toc533625484)

[4.2.2. Сортировка вставками 8](#_Toc533625485)

[4.2.3. Пузырьковая сортировка 10](#_Toc533625486)

[4.2.4. Сортировка подсчетом 11](#_Toc533625487)

[4.2.5. Быстрая сортировка 12](#_Toc533625488)

[4.2.6. Сортировка слиянием 12](#_Toc533625489)

[4.3. Описание функций 13](#_Toc533625490)

[Заключение: 17](#_Toc533625491)

[Литература: 18](#_Toc533625493)

# Введение:

Что делать с папкой в которой файлы не структурированы по какому-то определенному принципу? Нужно отсортировать в ней данные так чтобы было, чтобы наши данные имели такой вид с которым бы нам было приятно и понятно работать. Для этого необходим файловый менеджер, который по определенной характеристике отсортирует нам набор файлов в нужном нам порядке. В данном случае по возрастанию размера элемента в папке.

В программе представлены несколько видов сортировок: Сортировка выбором, Сортировка вставками, Пузырьковая сортировка, Сортировка подсчетом, Сортировка слиянием и Быстрая сортировка, с помощью которых пользователь сможет создать упорядоченную структуру в своей папке. Все они имеют большие различия между собой и представлены для того, чтобы пользователь мог выбрать наиболее подходящую для него.

# Постановка задачи:

***Постановка задачи:***

Разработать файловый менеджер с функцией показа файлов в заданном каталоге, упорядоченных по возрастанию/убыванию размера. Программа должана реализовать диалог с пользователем посредством интерфейса, который включает

1. возможность ввода пути до заданного каталога
2. возможность выбора метода сортировки
3. возможность просмотра отсортированного списка файлов с указанием размера

***Входные данные:***

1. Путь до директория, в которой необходимо отсортировать содержимое.
2. Метод сортировки.

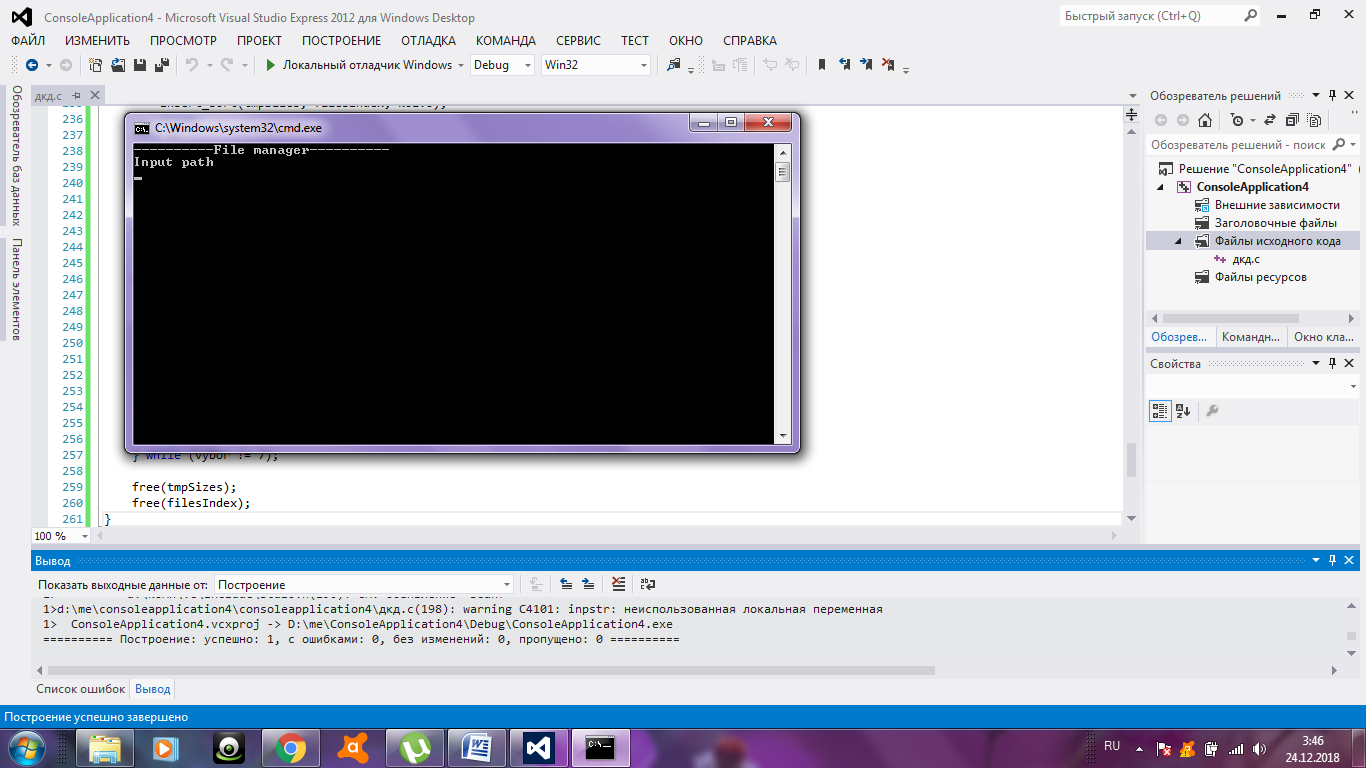
***Выходные данные:***

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

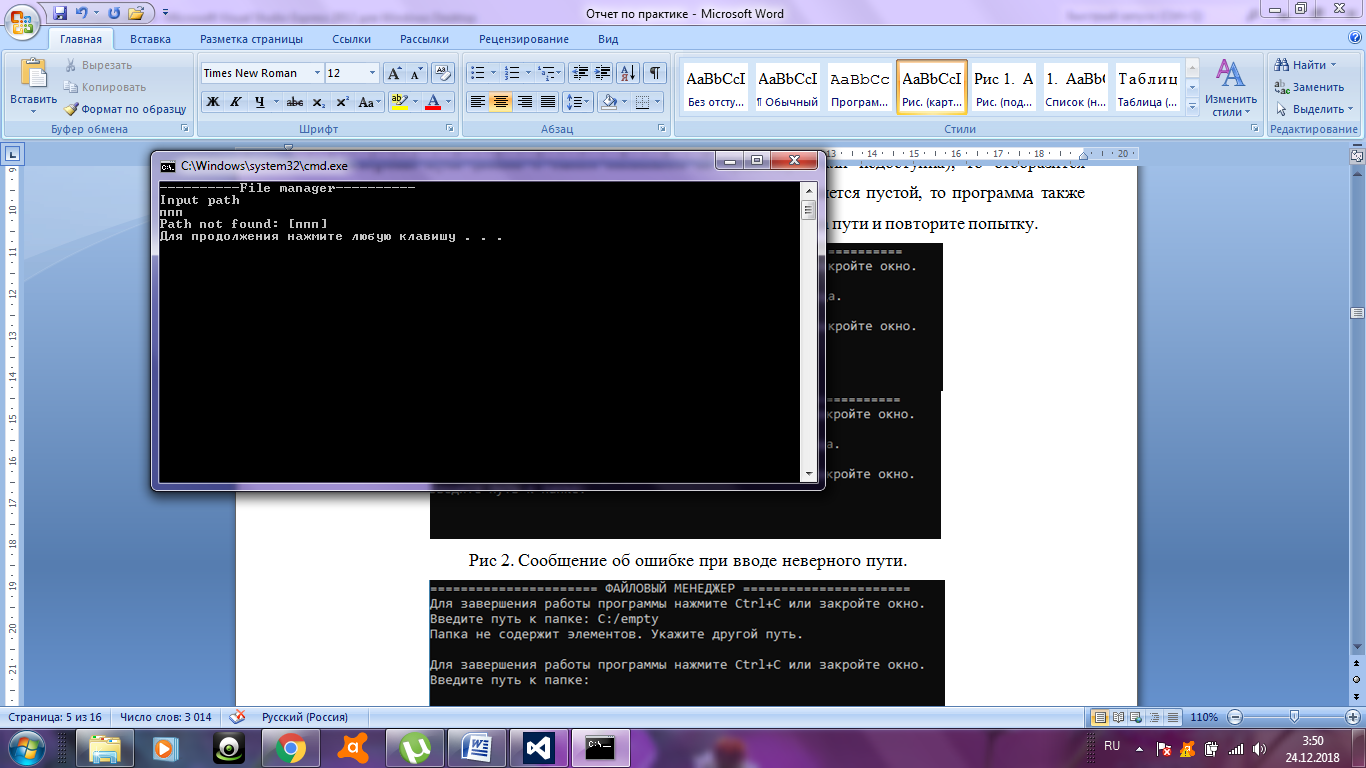
# Руководство пользователя

***В данном руководстве содержатся пошаговые инструкции по работе с программой:***

Запустите программу. Перед вами отобразится экран с предложением ввести путь к папке, из которой вы хотите отобразить файлы (см. Рис 1)

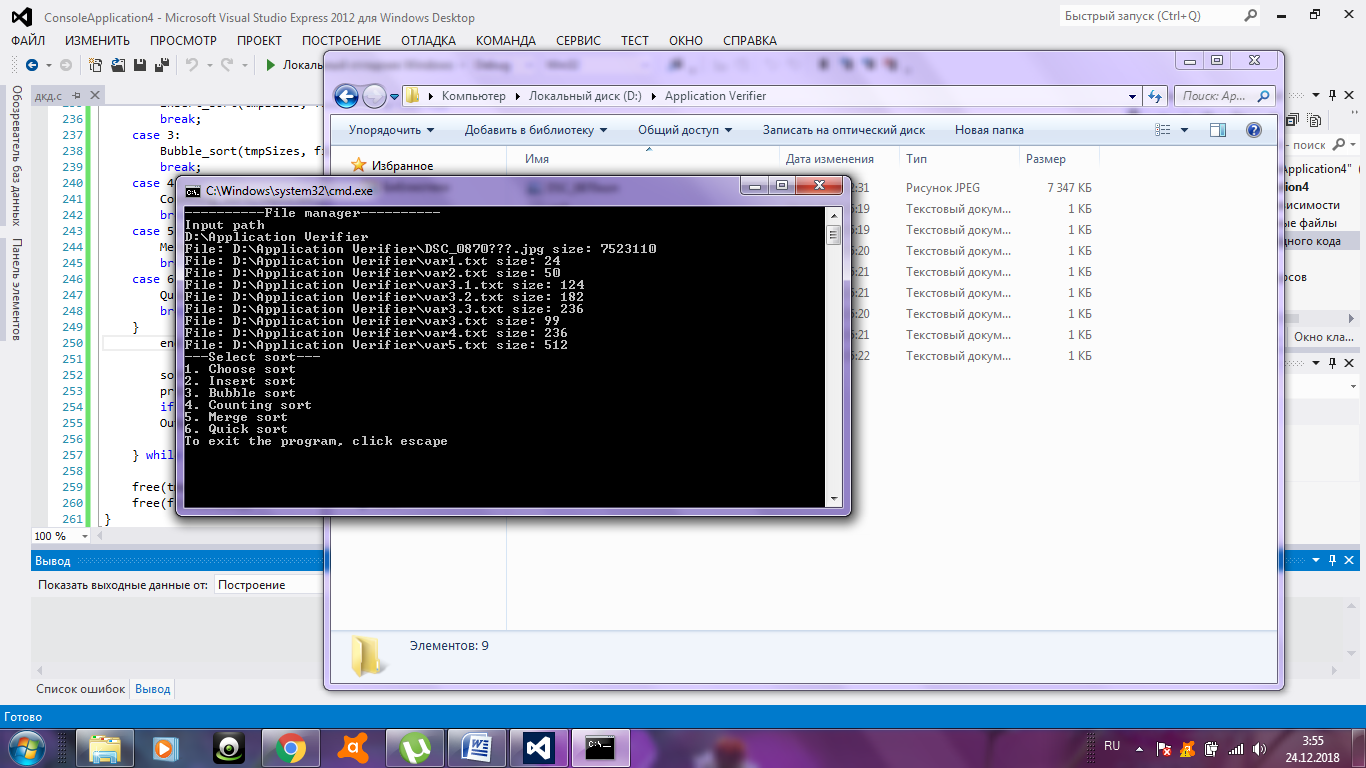


1. Программа после запуска.

С помощью клавиатуры введите полный путь к папке и нажмите Enter. Если вы указали неверный путь (попка с таким названием не существует или недоступна), то отобразится сообщение об ошибке (см. ). 

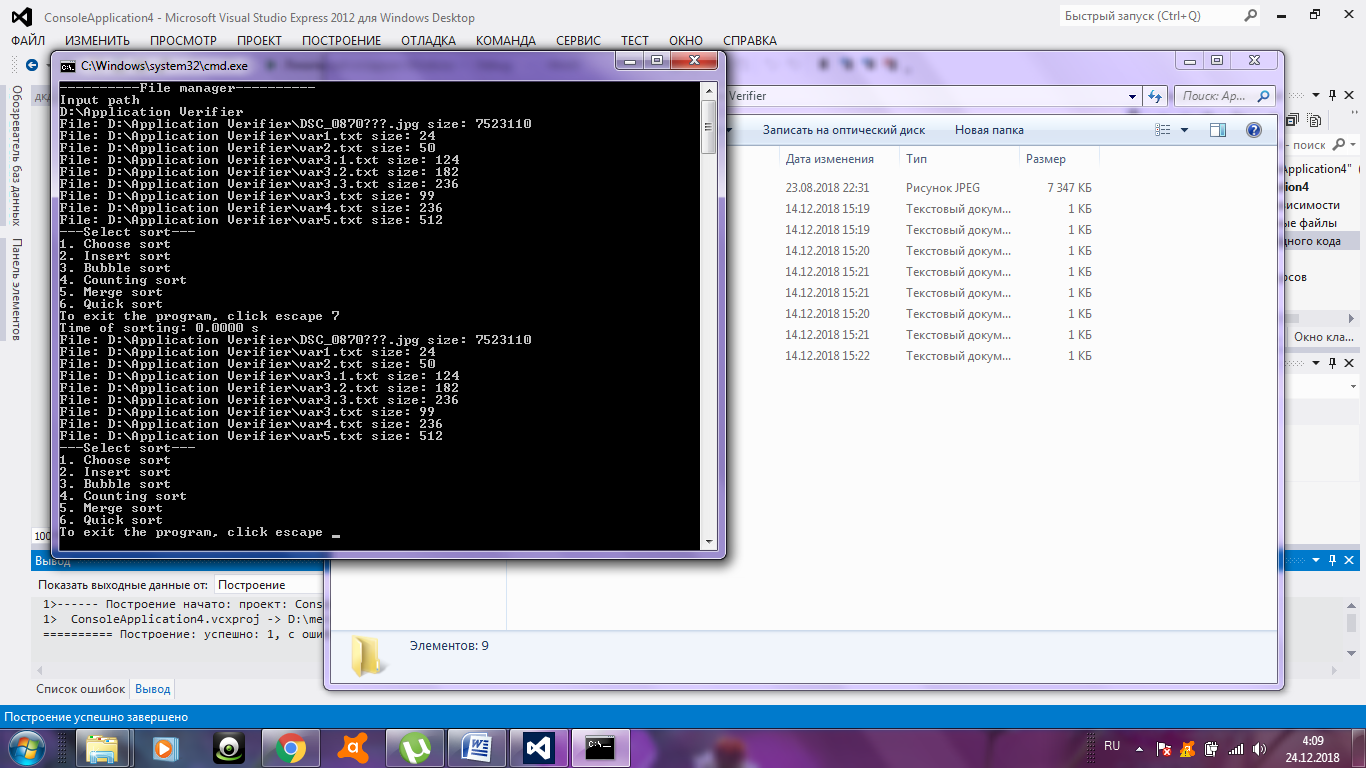
1. Сообщение об ошибке при вводе неверного пути.

При правильном вводе существующей папки отобразиться путь для каждого файла, его имя, тип и размер файла(см. Рис 3).



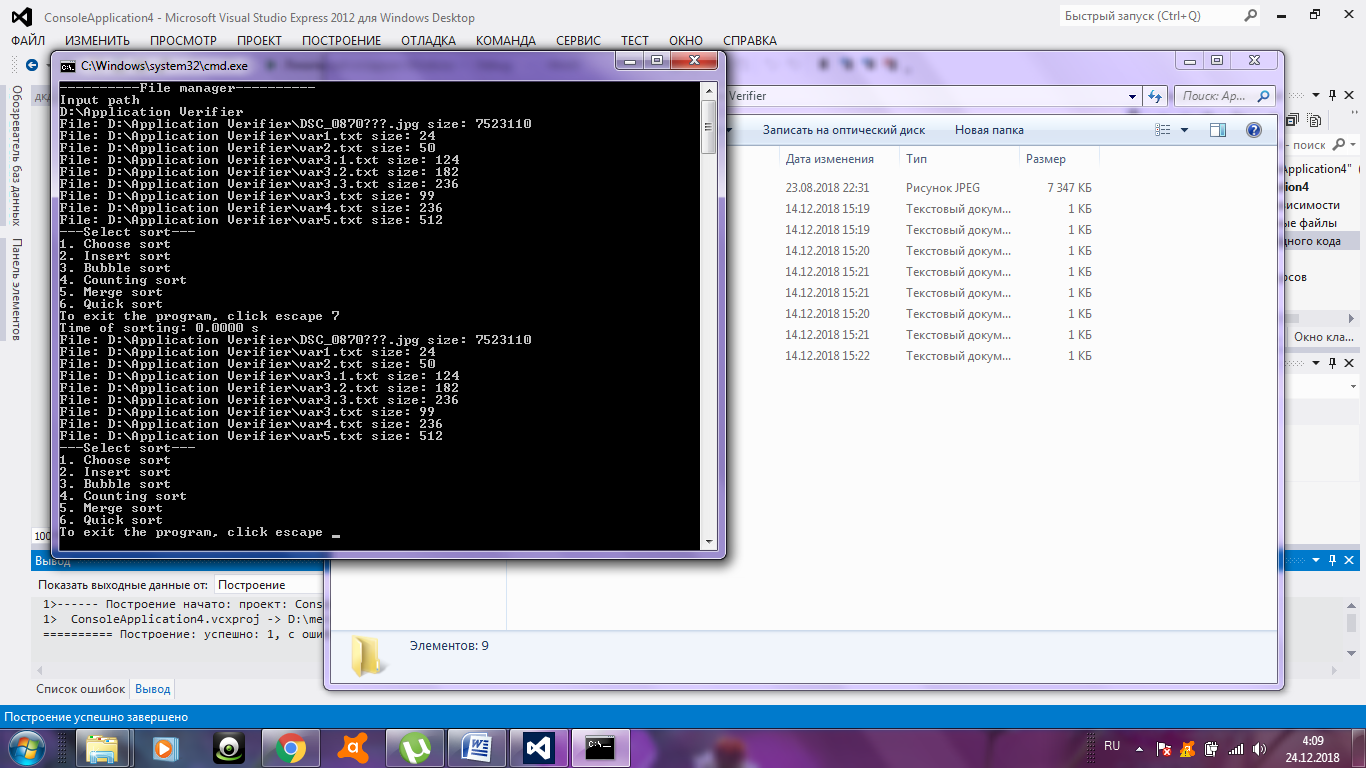
1. Первоначальный порядок файлов в папке.

Также сразу отобразится панель выбора методов сортировки файлов(см Рис 4)



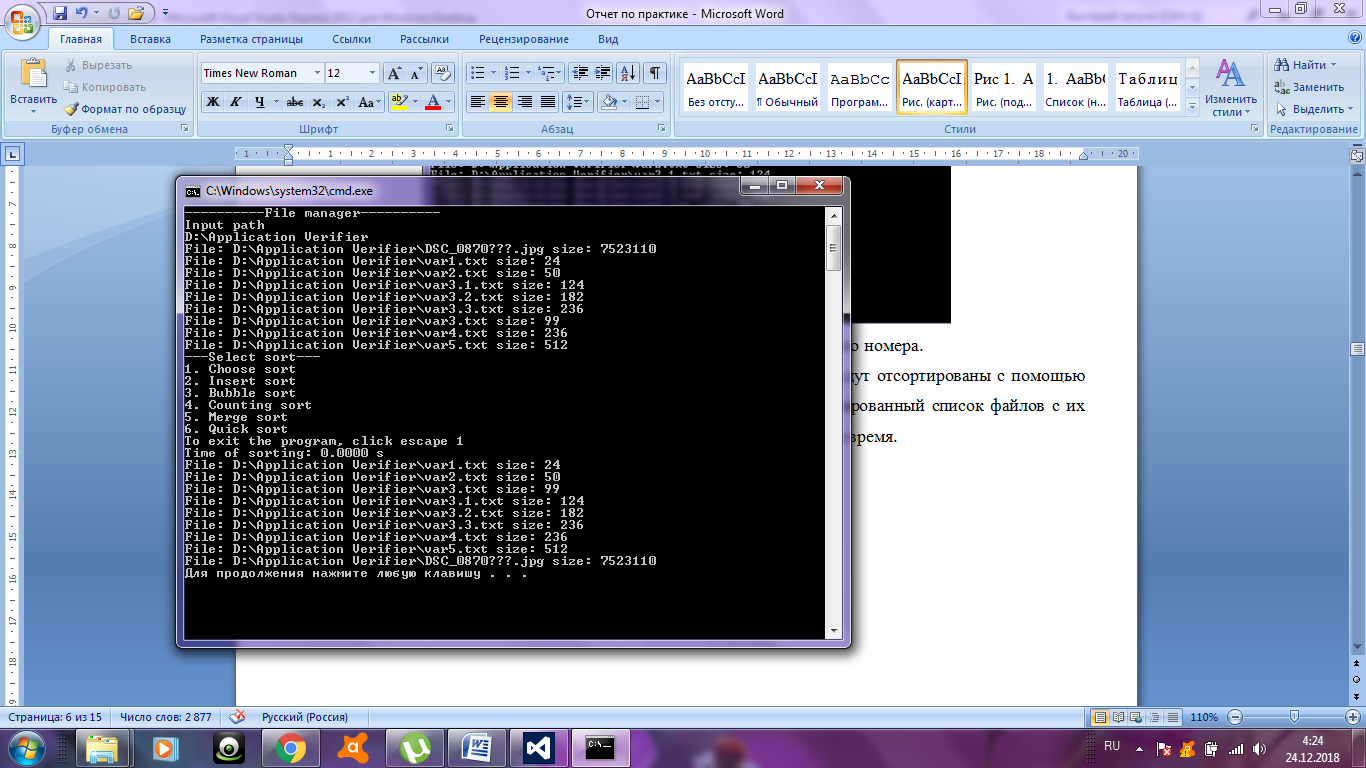
1. Панель выбора методов сортировки.

Введите цифру от 1 до 6, соответствующую выбранному вами методу, и нажмите Enter. Цифры указаны напротив их названий (для Сортировки подсчетом существуют ограничения для размера файла). Если вы ввели некорректный номер, то вам будет предложено повторить попытку ввода номера сортировки (см. Рис 5).



1. Экран программы после ввода неправильного номера.

Если номер метода сортировки указан верно, то далее файлы будут отсортированы с помощью выбранной сортировки. В результате программа выведет отсортированный список файлов с их путями, именами, типами и размерами, а также затраченное на это время.



1. Сортировка успешно завершена.

# Руководство программиста

## Структура программы

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

## Описание алгоритма

Файлы сортируются с помощью описанных ниже сортировок.

### Сортировка выбором

***Алгоритм сортировки выбором:***

1. В неотсортированном подмассиве ищется локальный максимум (минимум).
2. Найденный максимум (минимум) меняется местами с последним (первым) элементом в подмассиве.
3. Если в массиве остались неотсортированные подмассивы — смотри пункт 1.[1]
4. Пример сортировки выбором.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **5** | 2 | 4 | -3 | 0 | **-4** | 7 | -2 | 1 | 6 |
| **↑** |  |  |  |  | ↑ |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -4 | **2** | 4 | **-3** | 0 | 5 | 7 | 5 | 1 | 6 |
|  | ↑ |  | ↑ |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -4 | -3 | **4** | 2 | 0 | 5 | 7 | **-2** | 1 | 6 |
|  |  | ↑ |  |  |  |  | ↑ |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -4 | -3 | -2 | **2** | **0** | 5 | 7 | 4 | 1 | 6 |
|  |  |  | ↑ | ↑ |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -4 | -3 | -2 | 0 | **2** | 5 | 7 | 4 | **1** | 6 |
|  |  |  |  | ↑ |  |  |  | ↑ |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -4 | -3 | -2 | 0 | 1 | **5** | 7 | 4 | **2** | 6 |
|  |  |  |  |  | ↑ |  |  | ↑ |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -4 | -3 | -2 | 0 | 1 | 2 | **7** | **4** | 5 | 6 |
|  |  |  |  |  |  | ↑ | ↑ |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -4 | -3 | -2 | 0 | 1 | 2 | 4 | **7** | **5** | 6 |
|  |  |  |  |  |  |  | ↑ | ↑ |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -4 | -3 | -2 | 0 | 1 | 2 | 4 | 5 | **7** | **6** |
|  |  |  |  |  |  |  |  | ↑ | ↑ |

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

### Сортировка вставками

***Алгоритм сортировки вставками:***

1. Перебираются элементы в неотсортированной части массива.
2. Каждый элемент вставляется в отсортированную часть массива на то место, где он должен находиться.[2]
3. Пример сортировки вставками.

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

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

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

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

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

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

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

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

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

### Пузырьковая сортировка

***Алгоритм пузырьковой сортировки:***

Идем от начала массива, попарно сравнивая элементы. Если первый элемент больше второго, меняем их местами.

При достижении конца массива последний элемент считается полностью отсортированным.

Действия выше повторяем, двигаясь каждый раз от первого элемента до конца неотсортированного фрагмента массива до тех пор, пока он не будет полностью отсортирован.[3]

1. Пример пузырьковой сортировки.

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

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

### Сортировка подсчетом

***Алгоритм сортировки подсчетом:***

1. Для каждого элемента найти, сколько элементов, меньших определенного числа, и поместить это число на соответствующие место:
2. Делается это так:
3. За линейный проход по массиву мы для каждого из возможных значений подсчитываем, сколько элементов имеют такое значение.
4. Потом добавляем к каждому из найденных чисел суму всех предыдущих. Получая, таким образом, сколько есть элементов, значения которых не больше данного значения.
5. Далее, опять-таки за линейный проход, формируем из исходного массива новый отсортированный. При этом следим, чтобы два одинаковых элемента не были записаны в одно место.[4]
6. Пример сортировки подсчетом.

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

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| 1 | 3 | 4 | 2 |

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

### Быстрая сортировка

***Алгоритм быстрой сортировки:***

1. Массив разбивается на две части, с условием, что все элементы первой части меньше любого элемента второй.
2. Потом каждая часть сортируется отдельно. Разбиение на части достигается упорядочиванием относительно некоторого элемента массива, т. е. в первой части все числа меньше либо равны этому элементу, а во второй, соответственно, больше либо равны.
3. Два индекса проходят по массиву с разных сторон и ищут элементы, которые попали не в свою группу. Найдя такие элементы, их меняют местами.
4. Тот элемент, на котором индексы пересекутся, и определяет разбиение на группы.[5]
5. Пример быстрой сортировки.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 4 | 5 | 3 | -1 | 6 |
| ↑ | ↑↑ | ↑↑ | ↑ |  |
| -1 | 3 | 5 | 4 | 6 |
|  |  | ↑ | ↑ |  |
| -1 | 3 | 4 | 5 | 6 |

### Сортировка слиянием

***Алгоритм сортировки слиянием:***

1. Разбить массив на две части (разбиваем пока можем)
2. Отсортировать каждую из них, а потом слить обе части в одну отсортированную.[6]
3. Пример сортировки слиянием.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 5 | 0 | 5 | 3 | 7 |
| 2 | 5 | 0 | 5 | 3 | 7 |
| 2 | 5 | 0 | 3 | 5 | 7 |
| 0 | 2 | 5 | 3 | 5 | 7 |
| 0 | 2 | 3 | 5 | 5 | 7 |

## Описание функций

int ListDirectoryContents(const wchar\_t \*sDir, wchar\_t \*\*fileName, ULONGLONG \*filesSize)

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

***Входные параметры:* \***sDir – строка, содержащая путь к папке, \*\*fileName, \*fileSizes –массивы для записи полных имен и размеров файлов.

***Выходные параметры:***

j

void Input(wchar\_t \*\*sDir)

***Назначение:*** Выделение памяти под массив

***Входные параметры:*** \*\*sDir - строка, содержащая путь к папке.

***Выходные параметры:***

Отсутствуют

void Output(wchar\_t \*\*filesName, ULONGLONG \*filesSize, int \*filesIndex, int kolvo)

***Назначение:*** вывод содержимого каталога в виде списка файлов и их размеров в байтах.

***Входные параметры:***

\*\*filesName - указатель на массив имен,

\*filesSize – указатель на массив размеров,

\*filesIndex - упорядочивающий массив, содержащий последовательность индексов отсортированных элементов связанных массивов имен и размеров,, kolvo – количество файлов в директории

***Выходные параметры:***

Отсутствуют

void Choose\_sort(ULONGLONG \*tmpSize, int \*filesIndex, int kolvo)

***Назначение:***

Сортировка массива размеров файлов сортировкой выбором.

***Входные параметры:***

\* filesIndex – массив индексов массива размеров файлов

\* tmpSize – массив размеров файлов

kolvo – количество файлов в каталоге

***Выходные параметры:***

Отсутствуют

void Insert\_sort(ULONGLONG \*tmpSize, int \*filesIndex, int kolvo)

***Назначение:***

Сортировка массива размеров файлов сортировкой вставками.

***Входные параметры:***

\* filesIndex – массив индексов массива размеров файлов

\* tmpSize – массив размеров файлов

kolvo – количество файлов в каталоге

***Выходные параметры:***

Отсутствуют

void Bubble\_sort(ULONGLONG \*tmpSize, int \*filesIndex, int kolvo)

***Назначение:***

Сортировка массива размеров файлов пузырьковой сортировкой.

***Входные параметры:***

\*filesIndex – массив индексов массива размеров файлов

\* tmpSize – массив размеров файлов

kolvo – количество файлов в каталоге

***Выходные параметры:***

Отсутствуют

void Counting\_sort(ULONGLONG \*tmpSize, int \*filesIndex, int kolvo)

***Назначение:***

Сортировка массива размеров файлов сортировкой подсчетом.

***Входные параметры:***

\* filesIndex – массив индексов массива размеров файлов

\* tmpSize – массив размеров файлов

kolvo – количество файлов в каталоге

***Выходные параметры:***

Отсутствуют

void Merge(int \*filesIndex, ULONGLONG \*tmpSize, int ind1, int mid, int ind2)

***Назначение:***

Функция необходимая для сортировки слиянием. Соединяет подмассивы, упорядочивая их.

***Входные параметры:***

\* filesIndex – массив индексов массива размеров файлов

\* tmpSize – массив размеров файлов

ind1 – индекс первого элемента

mid – индекс среднего

ind2 – индекс последнего элемента

***Выходные параметры:***

Отсутствуют

void Merge\_sort(ULONGLONG \*tmpSize, int \*filesIndex, int ind1, int ind2)

***Назначение:***

Сортировка массива размеров файлов сортировкой слиянием.

***Входные параметры:***

\* filesIndex – массив индексов массива размеров файлов

\* tmpSize – массив размеров файлов

ind1 – индекс первого элемента

mid – индекс среднего

ind2 – индекс последнего элемента

***Выходные параметры:***

Отсутствуют

void Quicksplit(int \*filesIndex, ULONGLONG \*tmpSize, int \*i, int \*j, ULONGLONG opora)

Назначение

***Назначение:***

Функция необходимая для быстрой сортировки. Все элементы, меньшие опорного,перемещаются влево от него, большие – вправо.

***Входные параметры:***

\* filesIndex – массив индексов массива размеров файлов

\* tmpSize – массив размеров файлов

\*i – адрес первого индекса

\*j – адрес второго индекса

opora– опорный элемент

***Выходные параметры:***

Отсутствуют

void Quick\_sort(ULONGLONG \*tmpSize, int \*filesIndex, int ind1, int ind2)

***Назначение:***

Сортировка массива размеров файлов быстрой сортировкой

***Входные параметры:***

\* filesIndex – массив индексов массива размеров файлов

\* tmpSize – массив размеров файлов

ind1 – индекс первого элемента

ind2 – индекс последнего элемента

***Выходные параметры***:

Отсутствуют

void main()

***Назначение:*** основная функция (точка входа).

***Входные параметры:*** отсутствуют.

# Заключение:

Был разработан прототип файлового менеджера который выполняет сортировку файлов в директории с помощью 6 сортировок: Сортировка выбором, Сортировка вставками, Пузырьковая сортировка, Сортировка подсчетом, Сортировка слиянием и Быстрая сортировка. Во время использования программы легко заметить, что время реализации и объем использования памяти для каждой сортировки различна. Следующие таблицы показывают эффективность каждого метода.

1. Сложность методов сортировки[8]

|  |  |  |  |
| --- | --- | --- | --- |
| Методы сортировки | Средняя сложность алгоритма | | |
| T(max) | T(min) | V(max) |
| Сортировка выбором | O(n2) | O(n2) | O(1) |
| Сортировка вставками | O(n2) | O(n) | O(1) |
| Пузырьковая сортировка | O(n2) | O(n) | O(1) |
| Сортировка подсчетом | O(n) | O(n) | max(n) – min(n) + 1 |
| Быстрая сортировка | O(n2) | O(n·log2 n) | O(n) |
| Сортировка слиянием | O(n·log2 n) | O(n·log2 n) | O(n) |

1. Время реализации от количества файлов в директории[9]

|  |  |  |  |
| --- | --- | --- | --- |
| Методы сортировки | Время реализации от количества файлов в директории | | |
| 500 | 5000 | 10000 |
| Сортировка выбором | 0.001 | 0.143 | 0.545 |
| Сортировка вставками | 0.001 | 0.088 | 0.377 |
| Пузырьковая сортировка | 0.002 | 0.180 | 0.732 |
| Сортировка подсчетом | 0.002 | 0.022 | 0.043 |
| Быстрая сортировка | 0 | 0.002 | 0.005 |
| Сортировка слиянием | 0.001 | 0.037 | 0.134 |

## Из таблиц можно сделать вывод о том, что быстрая сортировка самая эффективная, а пузырьковая ­­­­­­­­­­­­­­­­– неэффективная.

# Литература:

1. http://www.c-cpp.ru/content/sortirovka-vyborom
2. http://kvodo.ru/sortirovka-vstavkami-2.html
3. http://algolist.manual.ru/sort/bubble\_sort.php
4. https://ru.stackoverflow.com/questions
5. https://prog-cpp.ru/sort-merge/
6. http://kvodo.ru/quicksort.html
7. https://tproger.ru/translations/sorting-for-beginners/
8. https://ru.wikipedia.org/wiki
9. https://habr.com/post/335920/