

**Министерство образования и науки Российской Федерации**  
**Федеральное государственное бюджетное образовательное учреждение**  
**высшего образования «Московский государственный университет технологий**  
**и управления имени К.Г. Разумовского (Первый казачий университет)».**  
**Университетский колледж информационных технологий**

Специальность 09.02.03 «Программирование в компьютерных системах»

## **КУРСОВОЙ ПРОЕКТ**

Модуль ПМ.01 Разработка программных модулей программного обеспечения  
для компьютерных систем

МДК.01.02 Прикладное программирование

на тему «Разработка файлового менеджера»

**Пояснительная записка**  
**УКИТ 09.02.03.2017.304.06ПЗ**

Группа

П-304к

Нормативный  
контроль

\_\_\_\_\_  
(личная подпись)

Гусева Е.Л.

Руководитель  
проекта

\_\_\_\_\_  
(личная подпись)

Глускер А. И.

Студент

\_\_\_\_\_  
(личная подпись)

Максимов И.А.

Москва 2017

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 ОСНОВНАЯ ЧАСТЬ.....	5
1.1 Введение в предметную область .....	5
1.2 Спецификация .....	5
1.3 Программа и методика испытания .....	8
1.4 Требования к программной документации .....	10
1.5 Средства и порядок испытания.....	11
1.6 Методы испытаний .....	12
1.7 Технический проект.....	16
1.8 Реализация программного изделия на языке программирования.....	20
1.9 Тестирование программного продукта .....	25
ЗАКЛЮЧЕНИЕ .....	27
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	28
ПРИЛОЖЕНИЕ А Руководство оператора .....	29
ПРИЛОЖЕНИЕ Б Исходный код.....	30
ПРИЛОЖЕНИЕ В Протокол системы контроля версий.....	39
ПРИЛОЖЕНИЕ Г Руководство программиста.....	40
ПРИЛОЖЕНИЕ Д Доклад.....	42

## ВВЕДЕНИЕ

Цель курсового проекта – разработать файловый менеджер, который позволит управлять файлами в операционной системе.

Основные задачи, которые были поставлены при разработке файлового менеджера:

- Анализ предметной области;
- Разработка технического задания;
- Проектирование файлового менеджера;
- Разработка файлового менеджера;
- Тестирование файлового менеджера.

Структура курсового проекта:

В каждом разделе должно быть введение и заключение. В этих разделах описывается текущий план работы по разработке файлового менеджера.

Далее идет основная часть, в которой идет сама разработка файлового менеджера. После разработки разрабатывается спецификация, в которой описывается техническое задание, устанавливающее требование к разрабатываемой программе. Следующим пунктом идет программа и методика испытаний, где описываются все методы и этапы тестирования программы. Также должны быть приложены тестовые примеры к разработанному приложению. После программы и методики испытаний следует описать технический проект программного продукта, в котором описываются архитектурные решения, которые применялись при разработке. Далее идет реализация и описание процесса разработки. После реализованного программного продукта идет его тестирование. Тестирование должно производиться по пунктам, описанных в программе и методике испытаний. После заключения, в котором будут описаны итоги по проделанной работе, последует раздел со списком использованной литературы. Там раскрываются источники, которые использовались при разработке файлового менеджера. Последним разделом является приложения, которые приводят

дополнительную информацию, которая включает в себя руководство оператора, исходный код приложения, протокол системы контроля версий, руководство программиста и доклад.

# **1 ОСНОВНАЯ ЧАСТЬ**

## **1.1 Введение в предметную область**

Данный раздел описывает предметную область, которая представляет собой работу файлового менеджера. «Файловый менеджер – компьютерная программа, предоставляющая интерфейс пользователя для работы с файловой системой и файлами»<sup>1</sup>. Файловый менеджер должен представлять из себя интуитивно понятное приложение с возможностями управления файловой системой, под управлением Windows, Linux, Mac OS. Файловый менеджер должен осуществлять управление как мышью, так и при помощи клавиатуры.

## **1.2 Спецификация**

В данном разделе описана основная информация о программном продукте, требования по эксплуатации и разработке, а также выполняемый им функционал.

### **1.2.1 Наименование программы**

Кроссплатформенный файловый менеджер.

#### **1.2.1.2 Область применения**

Область работы требующая использование файловой системы операционной системы.

### **1.2.2 Основание для разработки**

#### **1.2.2.2 Документ, на основании которого ведётся разработка**

Техническое задание на курсовой проект.

#### **1.2.2.3 Наименование и (или) условное обозначение темы для разработки**

File Manager.

### **1.2.3 Назначение разработки**

#### **1.2.3.2 Функциональное назначение**

Обеспечение свободной работы пользователя с файловой системой операционной системы.

---

<sup>1</sup> [https://ru.wikipedia.org/wiki/Файловый\\_менеджер](https://ru.wikipedia.org/wiki/Файловый_менеджер)

### **1.2.3.3 Требования к программе или программному изделию**

#### **1.2.3.3.1 Типы пользователей**

- Администратор.

### **1.2.3.4 Требования к функциональным характеристикам**

#### **1.2.3.4.1 Требования к системе тестирования**

Тесты представляют собой утверждения, которые после выполнения, нужно проверять явно в любом другом проводнике (файловом менеджере) операционной системы.

#### **1.2.3.4.2 Требования к составу выполняемых функций**

Администратор имеет следующие возможности работы с файлами/каталогами:

- Переименование;
- Удаление;
- Копирование;
- Вырезание;
- Просмотр свойств.

#### **1.2.3.4.3 Требования к персоналу**

Минимальное кол-во персонала – один человек, владеющий русским языком и обладающий практическими навыками работы в простом приложении.

### **1.2.3.5 Требования к составу и параметрам технических средств**

В состав технических средств должен входить компьютер клиента, включающий:

- Процессор Pentium 4 или выше;
- Клавиатура и мышь;
- Видеокарта;
- Монитор;
- HDD объёмом 30 ГБ и более.

### **1.2.3.6 Требования к информационной и программной совместимости**

#### **1.2.3.6.1 Требования к информационным структурам на входе и выходе не предъявляются**

#### **1.2.3.6.2 Требования к методам решения**

Файл менеджер должен быть разработан с использованием следующий технологий:

- C++ 11;
- Фреймворк Qt Creator;
- Инструмент Qt Designer;
- Инструмент Doxygen;
- Библиотека Boost.

### **1.2.4 Требования к программной документации**

#### **1.2.4.2 Состав программной документации**

- Состав программной документации должен включать:
- Техническое задание;
- Пояснительная записка;
- Текст приложения;
- Программа и методика испытания;
- Руководство пользователя;
- Руководство программиста.

#### **1.2.4.3 Специальные требования к пояснительной записке**

Специальные требования не предъявляются.

#### **1.2.4.4 Требования к исходному коду**

- Исходные коды на C++ должны удовлетворят требованиям исходным кодам Google;

### **1.2.5 Стадии и этапы разработки**

#### **1.2.5.2 Стадии разработки**

Разработка осуществляется в три стадии:

- Техническое задание;

- Технический проект;
- Рабочий проект.

### **1.2.5.3 Этапы разработки**

- На стадии техническое задание осуществляется разработка, согласование и утверждение технического задания.
- На стадии технический проект осуществляется разработка, согласование и утверждение пояснительной записки.
- На стадии рабочий проект осуществляется разработка текста программы.

### **1.2.6 Порядок контроля и приёмки**

**1.2.6.2 Приёмосдаточные испытания должно проводиться в соответствии с программой и методикой испытаний, разработанной, согласованной и утверждённой не позднее 25 декабря 2017 года**

В данном разделе можно найти краткую информацию о разрабатываемом программном продукте, сроках работ, этапах разработки, особенности и прочее.

### **1.3 Программа и методика испытания**

Программа и методика испытаний предназначена для того, чтобы описать какие цели и задачи программный продукт должен решать. Здесь расписано полное описание выполняемых функций и способы проверки их работоспособности.

#### **1.3.1 Объект испытаний**

##### **1.3.1.1 Наименование**

Файловый менеджер.

##### **1.3.1.2 Область применения**

Деятельность по управлению файловой системой.

##### **1.3.1.3 Обозначение программы**

File Manager.

##### **1.3.1.4 Цель испытания**

Проверка соответствия программного продукта требованиям технического задания.



### **1.3.1.5 Требования к программе**

#### **1.3.1 Требования к функциональным характеристикам**

##### **1.3.1.1 Требования к составу выполняемых функций**

Программа должна обеспечивать выполнение следующих функций:

- Функция вывода двух областей работы;
- Функция навигации по программе;
- Функция Drag And Drop файлов между двумя областями;
- Функция открытия у файла/каталога;
- Функция удаления у файла/каталога;
- Функция копирования у файла/каталога;
- Функция перемещения у файла/каталога;
- Функция переименования у файла/каталога;
- Функция просмотра свойств у файла/каталога.

##### **1.3.1.2 Требования к организации выходных данных**

Выходные данные не требуются.

##### **1.3.1.2.1 Требования к организации входных данных**

Входные данные не требуются.

##### **1.3.1.3 Требования к временным характеристикам**

Время между обработкой события перемещения, копирования, вырезания, удаления, создания каталога должно вычисляться по формуле (1 мб = 0.5 сек).

#### **1.3.2 Требования к надежности**

Требуется проверять Drag And Drop файлов между областями работы. Также при создании каталога требуется проверять каталог на предмет создания. При переименовании должна производиться проверка на совершение функции.

#### **1.3.3 Требования к информационной и программной совместимости**

Программа должна работать под управлением операционной системы Windows 7 (п. 4.5.5 технического задания).

## **1.4 Требования к программной документации**

### **1.4.1 Состав программной документации**

Состав программной документации должен включать:

- Техническое задание;
- Инсталлятор;
- Руководство пользователя;
- Пояснительную записку;
- Текст программы;
- Текст программы, осуществляющий автоматическое тестирование

программы «Файловый менеджер»;

- Программу и методику испытаний.

### **1.4.2 Специальные требования к пояснительной записке**

Пояснительная записка должна содержать UML-схему.

### **1.4.3 Специальные требования к тексту программы**

#### **1.4.3.1 Требования к исходным кодам изложены в документе**

А. И. Глускер «Сборник задач по курсу “Основы алгоритмизации и программирования” [Электронный ресурс] – 2011/раздел 3.1

**1.4.3.2 Программа должна быть написана на языке C/C++ и компилироваться транслятором Qt Creator 4.3.3 (п. 4.5.4 технического задания)**

### **1.4.4 Специальные требования к Инсталлятору**

**1.4.4.1 Инсталлятор должен быть совместим по версии ОС с основной программой.**

#### **1.4.4.2 Этапы инсталлятора**

- 1 Соглашение с условиями пользования программы;
- 2 Выбор пути для установки программы;
- 3 Установка программы;
- 4 Уведомление об завершении установки программы.

### **1.4.2.3 Дополнительная информация об инсталляторе**

Каждый этап установки должен переходить к следующему через кнопку «Далее».

### **1.4.5. Специальные требования к Руководству пользователя**

#### **1.4.5.1 Специальные требования к ГОСТ**

Руководство пользователя должно быть оформлено по ГОСТ 19.505-79.

#### **1.4.5.2 Специальные требования к фотоотчету**

Прилагаемые скриншоты к руководству пользователя не должны превышать размеры: 1280x720, а также должны относиться исключительно к теме программы.

#### **1.4.5.3 Специальные требования к функциям**

Все функции должны быть подробно описаны.

Количество функций и их работу можно найти в методике испытаний программного продукта.

### **1.5 Средства и порядок испытания**

#### **1.5.1 Технические средства, используемые при проведении испытаний**

В состав технических средств IBM-совместимый компьютер, включающий:

- Процессор Intel Pentium или совместимый с ним и выше;
- Клавиатуру и/или мышь;
- Видеокарту;
- Монитор;
- Жесткий или SSD-диск на 30гб минимум.

#### **1.5.2 Программные средства, используемые при проведении испытаний**

В состав программных средств входит:

- Лицензионная копия операционной системы Windows, Linux, Mac OS (любой версии);
- Программа файловый менеджер.

### **1.5.3 Порядок проведения испытаний**

**1.5.3.1 Подготовка к проведению испытаний заключается в обеспечении наличия компьютера и программных средств, установленных на этом компьютере**

**1.5.3.2 Ход проведения испытаний документируется в протоколе, где указывается перечень проводимых испытаний, результат каждого испытания и возможно замечания.**

#### **1.5.3.3 Состав испытаний**

##### **1.5.3.3.1 Проверка состава программной документации**

##### **1.5.3.3.2 Проверка требований к программе**

Проверка обеспечения требований к программе в соответствии с методами описанными в программе и методике испытаний.

##### **1.5.3.3.3 Проверка требований к программной документации**

###### **1.5.3.3.3.1 Проверка пояснительной записки в соответствии с методом**

###### **1.5.3.3.3.2 Проверка текстов программ в соответствии с методом**

### **1.6 Методы испытаний**

#### **1.6.1 Метод проверки требований к программе**

Проверка осуществляется путем запуска функций (смотреть пункты в техническом задании) и сравнения результатов их запуска с ожидаемыми. Задачу осложняет разные конфигурации персональных компьютеров, возможны осложнения в зависимости от сборки.

##### **– Проверка областей работы**

Обе области работы должны выглядеть идентично (при первом запуске программы); Дерево в области работы должно соответствовать корневому каталогу и быть идентичным корневому каталогу в проводнике Windows;

##### **– Проверка наличия дисков**

Обе области работы должны иметь выше одинаковое количество дисков. Если у диска размер свободного пространства равен нулю, то вкладка должна быть не активной; иначе вкладка должна быть активна, и поддерживать все остальные функции;

- Проверка переименования

При переименовании каталога/файла требуется проверить совершилось ли указанное действие над исходным файлом в проводнике Windows; иначе должно открыться диалоговое окно с ошибкой;

- Проверка удаления

При удалении каталога/файла требуется проверить совершилось ли указанное действие над исходным файлом в проводнике Windows; иначе должно открыться диалоговое окно с ошибкой;

- Проверка создания каталога

При создании каталога требуется проверить совершилось ли указанное действие над исходным файлом в проводнике Windows; иначе должно открыться диалоговое окно с ошибкой;

- Проверка копирования

При копировании каталога/файла требуется проверить совершилось ли указанное действие над исходным файлом в проводнике Windows; иначе должно открыться диалоговое окно с ошибкой;

- Проверка вырезания файла/каталога

При вырезании каталога/файла требуется проверить совершилось ли указанное действие над исходным файлом в проводнике Windows; иначе должно открыться диалоговое окно с ошибкой;

- Проверка вставки файла/каталога

При вставке каталога/файла требуется проверить совершилось ли указанное действие над исходным файлом в проводнике Windows; иначе должно открыться диалоговое окно с ошибкой;

- Проверка наименования пути в адресной строке

При выделении файла или каталога требуется проверить правильность пути в адресной строке;

- Проверка наименования файла в строке названия файла

При выделении файла или каталога требуется проверить правильность наименования файла в строке состояния внизу;

- Проверка Drag and Drop

Необходимо проверить корректность функции Drag And Drop между двумя областями работы и исходный используемый файл в проводнике после;

- Проверка дисковых пространств

Необходимо проверить количество дисков в проводнике и сверить их во вкладках в рабочей области;

- Проверка диалогового окна свойств

Осуществляется сравнение атрибутов в окне свойства и в проводнике на предмет сходимости.

- Проверка кроссплатформенности на Linux

Осуществляется проверка программы на предмет идентичной работы между разными операционными системами; Все выше перечисленные функции должны функционировать на операционной системе Linux.

- Проверка кроссплатформенности на Windows

Осуществляется проверка программы на предмет идентичной работы между разными операционными системами; Все выше перечисленные функции должны функционировать на операционной системе Windows.

- Проверка кроссплатформенности на Mac OS

Осуществляется проверка программы на предмет идентичной работы между разными операционными системами; Все выше перечисленные функции должны функционировать на операционной системе Mac OS.

### **1.6.2 Метод проверки требований к составу программной документации**

Проверка состава программной документации осуществляется визуально путем сравнения набора предъявленных документов (в форме распечатки или в рукописной форме) списку. При этом исходные тексты программ должны быть предоставлены так же и в электронной форме.

В случае если набор предъявленных документов соответствует списку, а исходные тексты предоставлены также в электронной форме, то в протокол заносится запись: «Состав программной документации» – соответствует; в противном случае: «Состав программной документации» – не соответствует.

### **1.6.3 Метод проверки требований к пояснительной записке**

### **1.6.4 Методика проверки требований к исходным кодам**

#### **1.6.4.1 Требования к исходному коду Google C++ Style Guide**

##### **1.6.4.2 Общие правила именования**

Имена должны быть описательными; избегать сокращения.

Параметры шаблона должны соответствовать стилю именования для их категории: параметры шаблона типа должны соответствовать правилам для имен типов, а параметры шаблона непигового типа должны соответствовать правилам для имен переменных.

##### **1.6.4.2.1 Общие требования к наименованию переменных**

Имена переменных (включая функциональные параметры) и членов данных имеют строчные буквы, с подчеркиваниями между словами. Элементы данных классов (но не структур) дополнительно имеют завершающие символы подчеркивания. Например: `a_local_variable`, `a_struct_data_member`, `a_class_data_member_`.

##### **1.6.4.2.2 Общие требования к наименованию констант**

Переменные, объявленные `constexpr` или `const`, и значение которых фиксировано для продолжительности программы, называются с ведущим «k», за которым следует смешанный случай. Например: `const int kDaysInAWeek = 7;`

##### **1.6.4.2.3 Общие требования к наименованию функций**

Регулярные функции имеют смешанный регистр; аксессоры и мутаторы могут быть названы как переменные.

##### **1.6.4.2.4 Общие правила именования файлов**

Имена файлов должны быть строчными и содержать символы подчеркивания (`_`) или тире (`-`). Следует использовать соглашение, которое использует проект. Если нет последовательного локального шаблона, следует выбрать «`_`».

##### **1.6.4.3 Общие требования к членам данных класса**

Элементы данных классов, как статические, так и нестатические, называются как обычные переменные, не являющиеся членами, но с завершающим подчеркиванием.

#### **1.6.4.4 Общие требования к комментариям**

Следует использовать или // , или /\* \*/ синтаксис.

#### **1.6.4.5 Общие требования к if-statement**

Не предпочитаются пробелы в круглых скобках. Ключевые слова if и else принадлежат отдельным строкам. Короткие условные операторы могут быть написаны на одной строке, если это повышает читаемость.

#### **1.6.4.6 Общие требования к циклам**

Скобки необязательны для циклов с одним заявлением.

### **1.6.5 Методика проверки инсталлятора**

После пройденных этапов, следует пройти по пути установленной программы и проверить наличие программы и всех сопутствующих файлов.

### **1.6.6 Методика проверки Руководства пользователя**

Каждая функция должна быть полностью описана. Все приложенные скриншоты должны отражать суть прилагаемого текста. Все скриншоты должны иметь размеры не более 1280x720 пикселей. Текст руководства пользователя должен быть оформлен по ГОСТ 19.505-79.

В этом разделе были изложены методы, которые будут применяться к файловому менеджеру для проверки соответствия требованиям технического задания. Эти данные позже будут занесены в таблицу отчетности.

## **1.7 Технический проект**

### **1.7.1.1 Наименование программы**

“Файловый менеджер”

### **1.7.1.2 Область применения**

Деятельность по управлению файловой системой.

### **1.7.1.3 Объект, в котором используют программу**

Программа «файловый менеджер» может быть использована на любых объектах.



#### **1.7.1.4. Основание для разработки**

##### **1.7.1.4.1 Документ, на основании которого ведется разработка**

Техническое задание на курсовой проект.

##### **1.7.1.4.2 Наименование и (или) условное обозначение темы для разработки**

File-Manager.

#### **1.7.1.5. Назначение разработки**

##### **1.7.1.5.1 Функциональное назначение**

Управление файловой системы.

##### **1.7.1.5.2 Эксплуатационное назначение**

Программа предназначена для использования неограниченным кругом лиц.

#### **1.7.2 Требования к программе или программному изделию**

##### **1.7.2.1 Требования к функциональным характеристикам**

##### **1.7.2.3 Требования к составу выполняемых функций**

Программа должна обеспечивать выполнение следующих функций:

- Функция вывода двух областей работы
- Функция навигации по программе
- Функция drag and drop файлов между двумя областями
- Функция открытия, удаления, копирования, перемещения, переименования файлов и создание каталога;
- Функция просмотра свойств у файла/каталога.

##### **1.7.2.4 Требования к пользовательскому интерфейсу**

Пользовательский интерфейс должен иметь динамическое разрешение.

Программа состоит из двух идентичных областей работы. В область работы входит:

- Главное окно с файловым менеджером;
- Над главным окном находится адресная строка, в которой содержится абсолютный путь до выделенного файла;
- Под главным окном файлового менеджера находится строка с наименованием выделенного файла;

- Главное окно файлового менеджера имеет вкладки сверху. Там отображены активные пространства. Если у диска свободное пространство равно нулю, вкладка не активна. Между вкладками можно переключаться.

#### **1.7.2.5 Требования к временным характеристикам**

Все функции определяются временем работы по формуле (1мб = 0.8сек).

#### **1.7.2.6 Требования к надежности**

##### **1.7.2.6.1 Другие требования к надежности не предъявляются**

#### **1.7.3 Условия эксплуатации**

##### **1.7.3.1 Минимальное количество персонала**

Один человек, владеющий русским языком, обладающий практическими навыками по использованию программ.

##### **1.7.3.2 Другие специальные требования к условиям эксплуатации не предъявляются**

##### **1.7.3.3 Требования к составу и параметрам технических средств**

В состав технический средств должен входить компьютер, включающий:

- процессор Intel Pentium или совместимый с ним;
- клавиатуру и/или мышь;
- видеокарту;
- монитор;
- жесткий или SSD-диск на 30гб минимум.

##### **1.7.3.4 Требования к информационной и программной совместимости**

##### **1.7.3.5 Требования к информационным структурам на входе и выходе не предъявляются**

##### **1.7.3.6 Программа должна быть написана на языке C++ и компилироваться в трансляторе Qt Creator 4.3.0 или выше.**

##### **1.7.3.7 Требования к исходному коду**

Исходный код должен быть исполнен по стандарту Google C++ Style Guide.

**1.7.3.8 Программа должна работать под управлением операционной системы Windows, Linux или Mac OS.**

**1.7.3.9 Требования к защите информации и программ не предъявляются**

**1.7.4 Требования к программной документации**

**1.7.4.1 Состав программной документации**

- Состав программной документации должен включать:
- Техническое задание;
- Пояснительную записку;
- Текст программы;
- Текст программы, осуществляющей автоматическое тестирование программы «Файловый менеджер»;
- Программу и методику испытаний.
- Руководство пользователя
- Инсталлятор

**1.7.4.2 Специальные требования к пояснительной записке**

Пояснительная записка должна содержать блок-схему (блок-схемы) алгоритма(-ов), используемых в программе стандарта UML.

**1.7.5 Стадии и этапы разработки**

**1.7.5.1 Стадии разработки**

Разработка осуществляется в три стадии:

- техническое задание;
- технический проект;
- рабочий проект.

**1.7.5.2 Этапы разработки**

На стадии техническое задание осуществляется разработка, согласование и утверждение технического задания в срок до 25 сентября 2017 года Глускером А. И.

На стадии технический проект осуществляется разработка, согласование и утверждение пояснительной записки в срок до 2 октября 2017 года Глускером А. И.

На стадии рабочий проект осуществляется разработка текста программы, осуществляющей автоматическое тестирование программы «Файловый менеджер», разработка, согласование и утверждение программы и методики испытаний, текста программы в срок до 20 декабря 2017 года Глускером А. И., после чего осуществляются испытания по результатам которой возможно будет проводиться корректировка программной документации в срок до 31 декабря 2017 года.

### 1.7.6 Порядок контроля и приемки

Приемосдаточные испытания должны проводиться в соответствии с программой и методикой испытаний, разработанной, согласованной и утвержденной не позднее 31 декабря 2017 год.

## 1.8 Реализация программного изделия на языке программирования

В этом разделе описывается разработка файлового менеджера.

### 1.8.2 Процесс реализации

Оригинал статей можно найти на сайте [Makasinov.ru/blog](http://Makasinov.ru/blog).

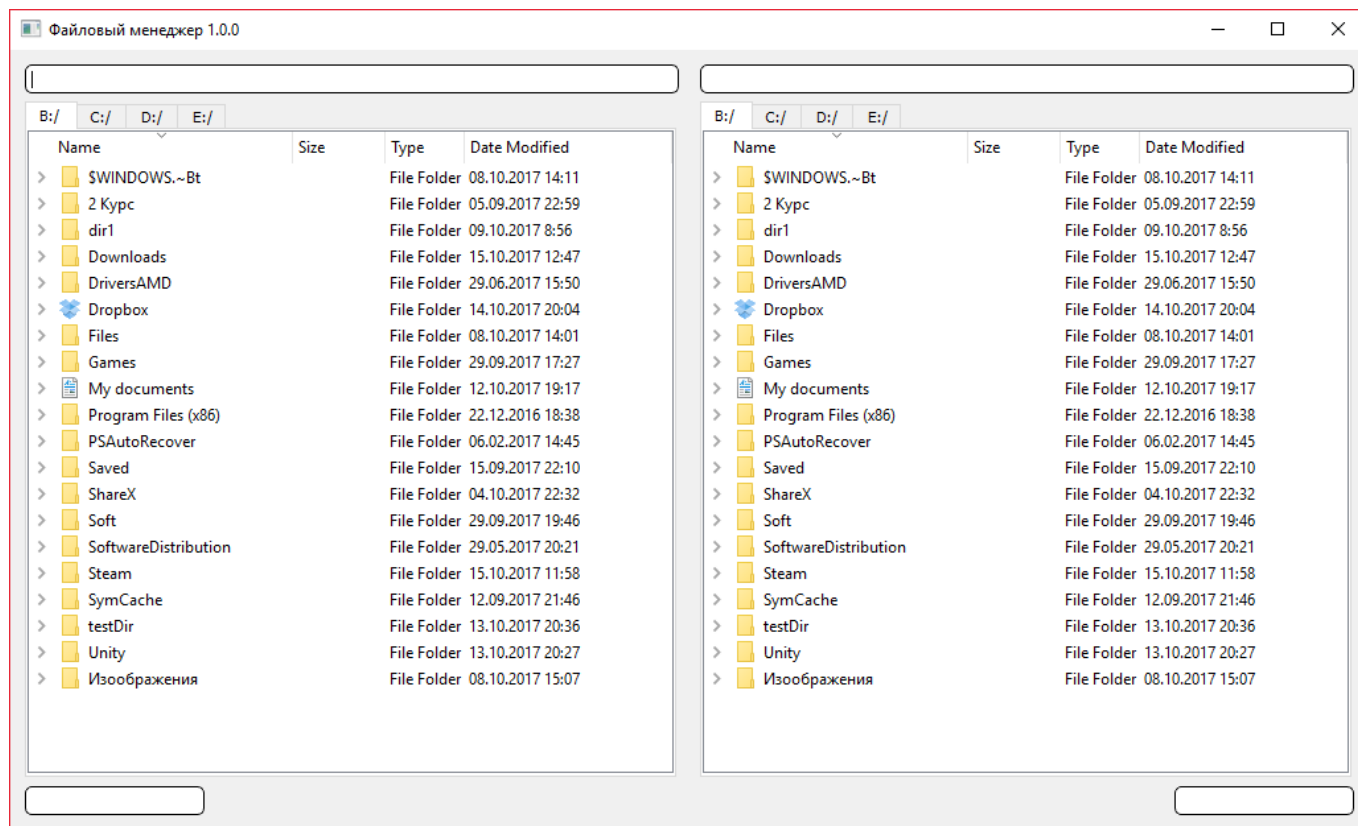


Рисунок 1 - Файловый менеджер

Разработка файлового менеджера началась в 15 сентября 2017 года. Во время работы использовалась программа Git для контроля версий программы. Сперва была

сделана GUI форма для работы с TreeView, Buttons и LineEdit'ами. Изначально в форме была лишь одна область работы.

Позже был добавлен сплиттер, разделяющий две области работы с файлами, и программа стала более походить на файловый менеджер (Рисунок 1). Обе области идентичны и работают одинаково.

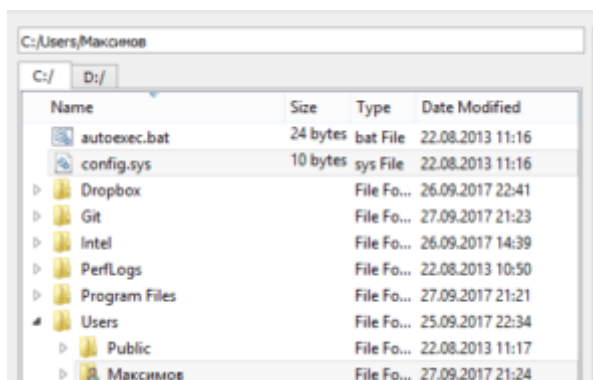


Рисунок 2 – Область работы 1

Сверху видны вкладки с дисками C:/ и D:/, они добавляются динамически в зависимости от того, сколько дисков задействовано (Рисунок 2).

После того как GUI форма была спроектирована, было реализовано соединение объектов с событиями. Так как файловые деревья добавлялись динамически, то и соединять приходилось каждое вручную. У меня на руках есть книжка по разработке приложений в Qt, но она достаточно старая. В новой версии Qt Creator разработчики изменили подход к соединению объектов и их событий (сигналов). Не сильно, но помучиться пришлось. Задачу усложнило то, что все деревья находятся в массиве, из-за одного символа & пришлось перерыть всю документацию. Когда проблема была решена, можно было приступить непосредственно к обработке тех самых событий.

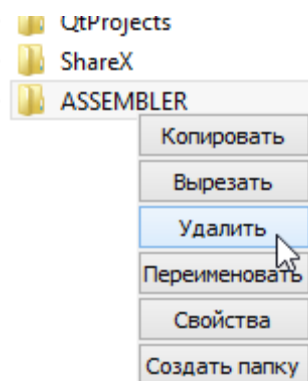


Рисунок 3 –  
Контекстное меню

Выпадающее меню с 6 пунктами было сделано вручную, без использования графического интерфейса (Рисунок 3). Генерируется динамически, когда пользователь нажимает правую кнопку мыши.

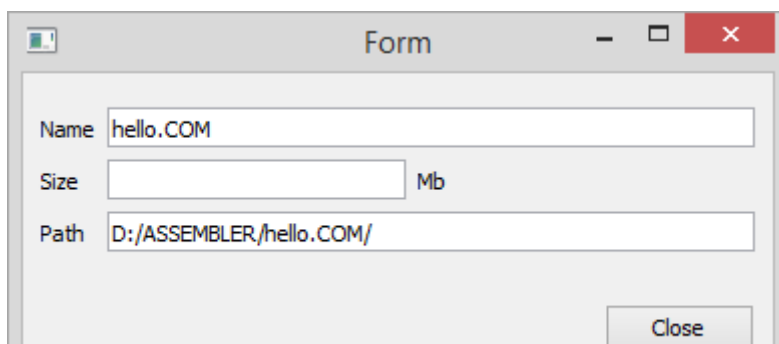


Рисунок 4 – Диалоговое окно свойств

Удаление работало, если у родительской ветви нет дочерних, иначе была бы ошибка удаления. Раньше я думал, что придётся удалять все подпапки рекурсивно, но позже выяснилось, что я просто использовал не тот метод.

Ранее по двойному щелчку происходило событие переименования файла/каталога, на момент релиза по двойному щелчку происходит открытие файла/каталога.

Интерфейс программы мне не очень нравился, он казался излишне строгим. Что навело меня на мысль немного сгладить углы у квадратных элементов. К счастью, в Qt можно изменить StyleSheet, что я и сделал.

Теперь внешний вид адресной строки и других элементов изменился на более визуально мягкий. Добавлена чёрная обводка и скругленные края.

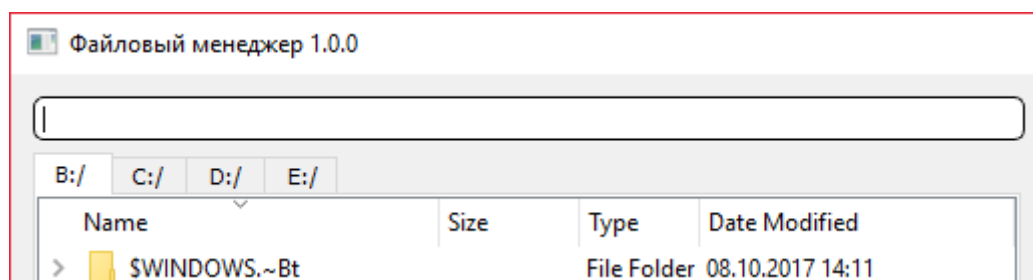


Рисунок 5 – Скругленное поле адреса файла/каталога

Изменилось и окно свойств.

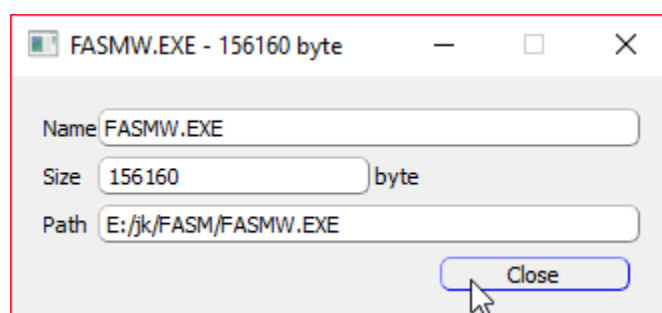


Рисунок 6 – Новое диалоговое окно свойств

Ранее PopUp меню в Qt, было реализовано через вертикальный ряд кнопок. Это не только ужасно смотрелось, но и затрудняло работу. Спасла Qt библиотека QMenu.

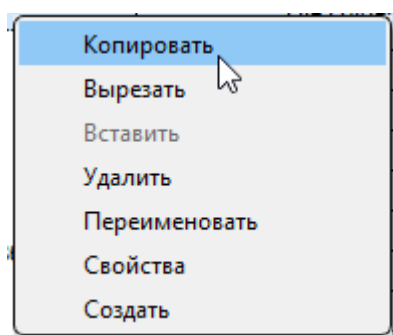


Рисунок 7 – Новое контекстное меню

Позднее были реализованы такие функции как: копирование, вырезание, вставка и удаление. Если при вставке будет ошибка (например, эту ошибку можно вызвать

путем вырезания и вставки объекта само в себя), то пользователь увидит соответствующее оповещающее диалоговое окно.

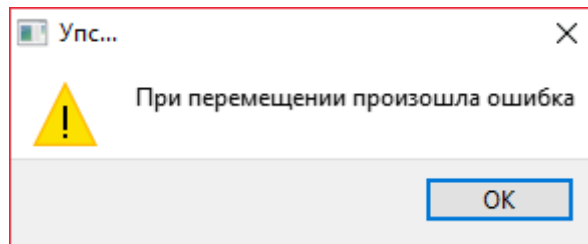


Рисунок 8 - Диалоговое окно с ошибкой

Больше недели не было решения, как быть с функциями удаления, копирования, вырезания и вставки. Но позже, было решено воспользоваться библиотекой `windows.h`. А для перевода из `QString` в `LPCWSTR` пришлось использовать интернет конструкцию.

Стоит отметить, что все функции в контекстном меню ориентированы не для одного элемента, а для всех выделенных.

Чтобы хоть как-то выделяться на фоне других файловых менеджеров, было решено добавить кроссплатформенность. Хотя фреймворк `Qt` позволяет осуществлять кроссплатформенные операции, но с файловой системой он работает плохо.

Сначала было решено использовать `C++ 17`, но `Qt` не позволил этого сделать. По умолчанию `Qt` использует `C++ 14` (на момент написания статьи), возможно позже перейдет на 17 версию.

Далее, после подключения `boost`'а и работая с алиасом `fs = boost::filesystem`, стало возможно использование таких методов как:

- `copy`, он же `copy_file`;
- `create_directory`;
- `remove` (использован `remove_all`);
- `rename`, для функции вырезать и вставить.

Одна проблема была обнаружена, когда встала потребность в осуществлении копировании/вырезании и вставки папки. Дело в том, что в `boost`'е нет такого метода как `copy_directory`, а если и есть аналоги типа простого `copy` или `copy_file`, то они



просто выдадут ошибку при использовании их на папке. К счастью, интернет уже наполнен решениями этой проблемы. Была реализована рекурсивную вставку, а в папке использованы итераторы, чтобы пройти по каждой «не папке» и сделать определенные операции.

Стоит уделить внимание на мелочь, которая ранее могла обрушить программу. Раньше программа определяла дисковод как за обычный диск, и вкладка сверху была активна. То есть можно было вполне попробовать записать файлы в пустой диск (дисковод), но была бы ошибка, которая бы фатальной для программы. Этот баг был исправлен и теперь, если приложение определяет пустой диск, то вкладка просто не активна. На неё можно переключиться, но сделать внутри ничего нельзя сделать.

В этом разделе был описан процесс разработки файлового менеджера.

### **1.9 Тестирование программного продукта**

В этом разделе идет подробное описание прохождения ранее описанных тестов.

#### **1.9.1 Тестирование исходного кода путём написания юнит-тестов не производилось.**

Сдача файлового менеджера происходит путём приёмо-сдаточных испытаний.

#### **1.9.2 Протоколы прохождения тестов**

Таблица 1 – Прохождение тестов

Тип проверки	Результат
Проверка областей работы	Успешно
Проверка наличия дисков	Успешно
Проверка переименования	Успешно
Проверка удаления	Успешно
Проверка создания каталога	Успешно
Проверка копирования	Успешно
Проверка вырезания файла/каталога	Успешно
Проверка вставки файла/каталога	Успешно
Проверка наименования пути в адресной строке	Успешно

Проверка наименования файла в строке названия файла	Успешно
Проверка drag and drop	Успешно
Проверка дисковых пространств	Успешно
Проверка диалогового окна свойств	Успешно
Проверка кроссплатформенности на Linux	<b>Не проверено</b>
Проверка кроссплатформенности на Windows	Успешно
Проверка кроссплатформенности на Mac OS	<b>Не проверено</b>

Большинство тестовых примеров были пройдены успешно. Не проверены пункты:

- Проверка кроссплатформенности на Linux
- Проверка кроссплатформенности на Mac OS

В данном разделе были описаны методы тестирования, которые использовались для проверки соответствия требованиям технического задания.

## ЗАКЛЮЧЕНИЕ

Результатом работы оказался разработанный файловый менеджер, который включает в себя следующее:

- Возможности удобной и комфортной работы с файловой системой любой операционной системы;
- Возможность просмотра свойств у файла/каталога;
- Возможность быстрой работы по копированию, вырезанию и вставки многих файлов путем встроенного Drag and Drop функционала;
- Возможность взаимодействия с файлами напрямую из файлового менеджера.

В итоге, было сделано заключение о том, что фреймворк на высоком уровне обеспечивает кроссплатформенность своих встроенных объектов, но для работы с файловой системы не годится. Для управления файловой системой операционных систем наилучшим решением будет являться библиотека boost.

Разработка подобных проектов требует аккуратности потому, что можно по неаккуратности удалить, изменить, или повредить файловую систему операционной системы.

Так же были закреплены навыки:

- Объектно-ориентированный анализ и проектирование;
- Оформление документации, удовлетворяющие стандартам ГОСТ;
- Программирование на C++;
- Программирование во фреймворке Qt;
- Изучена библиотека boost;
- Использование системы контроля версий.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Шлее М. Qt 4.8. Профессиональное программирование на C++. – СПб.: БХВ-Петербург, 2013. – 912 с.: ил. – (В подлиннике);
- 2 Официальный сайт с документацией Documents Qt io ( <http://doc.qt.io> ) ;
- 3 QT5 TUTORIAL MODELVIEW WITH QTREEVIEW AND QFILESYSTEMMODEL – 2017. bogotogo.com ( [http://www.bogotobogo.com/Qt/Qt5\\_QTreeView\\_QFileSystemModel\\_ModelView\\_MVC.php](http://www.bogotobogo.com/Qt/Qt5_QTreeView_QFileSystemModel_ModelView_MVC.php) ) ;
- 4 Qt/C++ - Урок 056. Подключение библиотеки Boost в Qt для компиляторов MinGW и MSVC с сайта evileg.com ( <https://evileg.com/ru/post/160> ) ;

# **ПРИЛОЖЕНИЕ А**

## **Руководство оператора**

### **1 Назначение программы**

#### **1.4 Наименование**

Файловый менеджер файловой системы операционной системы.

#### **1.5 Назначение**

Обеспечение комфортной работы с файлами и директориями в операционной системе.

#### **1.6 Характеристика области применения**

Файловый менеджер предназначен для использования любыми лицами, требующими использование файловой системы операционной системы.

### **2 Условия выполнения программы**

#### **2.2 Аппаратные средства**

В состав аппаратных средств пользователя должен входит компьютер, который включает в себя:

- Процессор;
- Видеокарта;
- Жёсткий диск;
- Сетевая карта;
- Клавиатура;
- Мышь.

#### **2.3 Программные средства**

В состав программных средств пользователя должны входить:

- Лицензионная копия операционной системы (Windows, Linux, Mac OS);
- Программа файловый менеджер.

### **3 Выполнение программы**

#### **4.1 Запуск программы**

Для запуска программы необходимо двойным щелчком на жать на исполняемый файл в установленной директории.

## ПРИЛОЖЕНИЕ Б

### Исходный код

#### Main.cpp

```
#include "mainwindow.h"
#include <QDragEnterEvent>
#include <QDropEvent>
#include <QApplication>
#include <QIcon>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.setWindowTitle("File Manager 1.1.3");

    w.setWindowIcon(QIcon("B:\\Dropbox\\newKursach\\icon.png"));
    w.show();

    return a.exec();
}
```

#### MainWindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include "customtree.h"
#include <QMainWindow>
#include <QFileSystemModel>
#include <QTreeView>
#include <QFileInfoList>
#include <QDebug>
#include <QSplitter>
#include <QModelIndex>
#include <windows.h>
#include <QKeyEvent>
#include <QDrag>
#include <QMimeData>
#include <QDragEnterEvent>
#include <QDropEvent>
#include <QDirModel>
#include <QTimer>
#include <QKeyEvent>
#include <QWidget>

namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT

public slots:
    bool event(QEvent *event);

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    bool empty = true;           // is "Clipboard" empty or not
    bool cutted = false;
    int selectedCount;
    QModelIndex selectedIndex;

private slots:
    void debug()
    { qDebug() << "DEBUG"; }

private:
    Ui::MainWindow *ui;

    QList<QString> * vec = new QList<QString>; // actually it's list,
                                              // but this name already
                                              // exists :3 (vector)
}
```

```

QFileInfoList list = QDir::drives();
int count = list.count();

QTreeView * tree = new QTreeView[count + count];
CustomTree * arr = new CustomTree[count + count];
QTimer timer;

};

#ifdef MAINWINDOW_H

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QStorageInfo>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    setFocusPolicy(Qt::StrongFocus);

    setAcceptDrops(true);

    ui->tabWidget ->removeTab(0);
    ui->tabWidget ->removeTab(0);
    ui->tabWidget_2->removeTab(0);
    ui->tabWidget_2->removeTab(0);

    QDirModel *model = new QDirModel;
    //QFileSystemModel *model = new QFileSystemModel;

    model->setFilter(QDir::Files|QDir::Dirs|QDir::NoDotAndDotDot); // Hide dots
    //model->setRootPath("");
    QDir::setCurrent(QDir::rootPath());
    model->setReadOnly(false);
    timer.start(500);

    for (int i = 0; i < count; ++i)
    {
        tree[i].setModel(model);
        tree[i].setRootIndex(model->index(list.at(i).path()));

        QObject::connect(&timer,SIGNAL(timeout()),
            &arr[i],SLOT(refreshTree()));

        arr[i].addVec(*vec);
        arr[i].addModel(model);
        arr[i].addTree(&tree[i]);
        arr[i].addPath(ui->addressEdit);
        arr[i].addOtherPath(ui->addressEdit2);
        arr[i].addFileName(ui->itemNameEdit);
        arr[i].addTab(ui->tabWidget);
        arr[i].addClipboard(this->selectedIndex);
        arr[i].addCount(this->selectedCount);
        arr[i].addBool(this->empty);
        arr[i].addCut(this->cutted);

        ui->tabWidget->addTab(&tree[i],list.at(i).path());

        QStorageInfo storage;
        storage.setPath(list.at(i).path());
        if ( storage.bytesAvailable()/1000/1000 == 0 )
            ui->tabWidget->widget(i)->setEnabled(false);
    }

    for (int i = 0, l = count; l < count + count; ++i, ++l)
    {
        tree[l].setModel(model);
        tree[l].setRootIndex(model->index(list.at(i).path()));

        QObject::connect(&timer,SIGNAL(timeout()),
            &arr[l],SLOT(refreshTree()));

        arr[l].addVec(*vec);
        arr[l].addModel(model);
        arr[l].addTree(&tree[l]);
    }
}

```

## Mainwindow.cpp

```

arr[l].addPath(ui->adressEdit2);
arr[l].addOtherPath(ui->adressEdit);
arr[l].addFileName(ui->itemNameEdit2);
arr[l].addTab(ui->tabWidget_2);
arr[l].addClipboard(this->selectedIndex);
arr[l].addCount(this->selectedCount);
arr[l].addBool(this->empty);
arr[l].addCut(this->cutted);

ui->tabWidget_2->addTab(&tree[l],list.at(i).path());

QStorageInfo storage;
storage.setPath(list.at(i).path());
if ( storage.bytesAvailable()/1000/1000 == 0 )
    ui->tabWidget_2->widget(i)->setEnabled(false);
}
}
bool MainWindow::event(QEvent * event)
{
    const auto CtrlPlusC = "\u0003";
    if ( event->type() == QEvent::KeyPress){
        QKeyEvent * keyEvent = static_cast<QKeyEvent *>(event);
        //qDebug() << keyEvent->text();
        for ( int i = 0; i < (count + count); i++ )
            if (tree[i].hasFocus() )
            {
                //if ( keyEvent->text()[5] == '3' );
                // qDebug() << keyEvent->text() << " event!";
                //else if ( ) qDebug() << "Delete event!"
                break;
            }
    };
    QMainWindow::event(event);
}
MainWindow::~MainWindow()
{ delete ui; }

```

## Customtree.h

```

#ifndef CUSTOMTREE_H
#define CUSTOMTREE_H

#include "prop.h"
#include <QTreeView>
#include <QStandardItem>
#include <QDesktopServices>
#include <QTableWidget>
#include <QLineEdit>
#include <QString>
#include <QObject>
#include <QMouseEvent>
#include <QDebug>
#include <QMenu>
#include <QAction>
#include <QSplitter>
#include <QFileInfo>
#include <QMessageBox>
#include <QDirModel>
#include <QInputDialog>
#include <QList>
#include <QKeyEvent>
#include <QTimer>
#include <boost/filesystem.hpp>
#include <iostream>
#include <string.h>

class CustomTree : public QObject
{
    Q_OBJECT
public:
    CustomTree(QObject *parent = nullptr);
    void addPath(QLineEdit * LineEdit) {this->adressLine = LineEdit;} //
    void addOtherPath(QLineEdit * LineEdit) {this->adressLine_2 = LineEdit;} //
    void addFileName(QLineEdit * LineEdit) {this->fileNameLine = LineEdit;} //
    void addTab(QTabWidget * tab) {this->tab = tab;} //
    void addTree(QTreeView * tree); //
    bool eraseDir(QModelIndex index); //
    void addModel(QDirModel * model) { this->model = model; } //
    void addBool(bool &EMPTY) {this->empty = &EMPTY;} //
    void addCut(bool &CUT) {this->cutted = &CUT;} //

```



```

void addClipboard(QModelIndex &SI) {this->selectedIndex = &SI;} //
void addCount(qint32 &SC) {this->selectedCount = &SC;} //
void addVec(QList<QString> &L) {this->vec = &L;} //
void eventHandle(QKeyEvent *event); //
bool scanDir(std::wstring oldFile, std::wstring newFile, boost::system::error_code error_code);
static const char slash = '/'; //
static const char nonSlash = '\\'; //

public slots:
void checkSelected();
void drawPath();
void openItem();
void popUp();
void popupCopy();
void popupCut();
void popupPaste();
void popupErase();
void popupRename();
void popupProp();
void popupMkdir();
void refreshTree() { this->model->refresh(); }
void debug() { qDebug() << "DEBUUUUG"; }

private:
bool * empty;
bool * cutted; // Cut action
int * selectedCount;
QModelIndex * selectedIndex;

QMenu * menu = new QMenu; // popUp menu with 5 fields
QAction * paste ;
QAction * copy ;
QAction * erase ; // not "delete", because it's engaged
QAction * cut ;
QAction * rename ;
QAction * mkdir ;
QAction * prop ;

QList<QString> * vec;
int itemsSelected;
Prop * propUi;
QTreeView * tree;
QDirModel * model;

QTabWidget * tab;
QLineEdit * adressLine; // absolute path to file
QLineEdit * adressLine_2; // second one
QLineEdit * fileNameLine;

};

#endif // CUSTOMTREE_H

#include "customtree.h"

/*
 * This function provides recursively copy the folder and files
 */
bool CustomTree::scanDir(std::wstring oldFile, std::wstring newFile, boost::system::error_code error_code)
{
    qDebug() << "scanDir - " << oldFile.data() << "\t" << newFile.data();
    boost::filesystem::create_directory(newFile.data(), error_code);
    //if ( error_code )
    //boost::filesystem::copy_directory(oldFile.toStdWString(), newFile.toStdWString(), error_code);

    boost::filesystem::path path(oldFile.data());

    boost::filesystem::directory_iterator end_itr;
    qDebug() << "Nu poehali - " << oldFile.data();
    for (boost::filesystem::directory_iterator itr(path); itr != end_itr; ++itr)
    {
        //if (boost::filesystem::is_regular_file(itr->path())) {
            std::wstring current_file = itr->path().wstring();
            std::wstring oldFile = newFile.data();
            oldFile += CustomTree::slash;
            std::wstring var = current_file.data();
            int position = var.length();

```

## Customtree.cpp

```

qDebug() << "-" << var.data();
while ( var[position] != CustomTree::nonSlash) --position;
var.erase(0,(position + 1));
qDebug() << "after - " << var.data() << "\t";
oldFile += var;
if ( boost::filesystem::is_directory(current_file.data()) )
    scanDir(current_file,oldFile,error_code);
else boost::filesystem::copy_file(current_file.data(),oldFile.data(),error_code) ;
qDebug() << "\t" << current_file.data() << " | " << oldFile.data() ;
//}
}
}

void CustomTree::popupPaste()
{
    checkSelected();
    if ( !vec->isEmpty() )
    {
        *empty = true;
        auto index = this->tree->currentIndex();

        QFileInfo info;
        if ( index.isValid() )
            info = this->model->fileInfo(index);
        else info = this->model->fileInfo(this->tree->rootIndex());
        auto it = vec->begin();
        while ( it != vec->end() )
        {
            QString oldFile = *(it++);
            QString newFile = info.absoluteFilePath();
            newFile += "/";
            newFile += *(it++);

            boost::system::error_code error_code;
            if ( boost::filesystem::is_directory(oldFile.toStdWString(),error_code) )
            {
                //qDebug() << "directory - " << oldFile;
                scanDir(oldFile.toStdWString(),newFile.toStdWString(),error_code);
                if ( *cutted ) {
                    boost::filesystem::remove_all(oldFile.toStdWString());
                    QString text = adressLine_2->text();
                    while ( *(text.end() - 1) != slash )
                    {
                        text.remove(-1,text.length());
                        //qDebug() << *text.end() << "\t" << text;
                    }
                    adressLine_2->setText(text);
                }
            } else {
                //qDebug() << "file - " << oldFile;
                boost::filesystem::copy_file(oldFile.toStdWString(),newFile.toStdWString(),error_code);
            }
            if ( error_code )
                QMessageBox::warning(this->tree, tr("Упс.."),tr("При копировании произошла ошибка"));
            //qDebug() << oldFile << " -> " << newFile;
        }
        this->model->refresh();
    }
}

void CustomTree::popupCut()
{
    checkSelected();
    this->popupCopy();
    *cutted = true;
}

void CustomTree::eventHandle(QKeyEvent *event)
{
    this->popupErase();
}

void CustomTree::popupCopy()
{
    checkSelected();
    if ( itemsSelected != 0)
    {
        *selectedIndex = this->tree->selectionModel()->selectedRows().at(0);
        *selectedCount = this->tree->selectionModel()->selectedRows().length();
        *empty = false; *cutted = false; vec->clear();
        QFileInfo info;
        for (int i = 0; i < *selectedCount ; i++)
    }
}

```

```

    {
        info = this->model->fileInfo(
            this->tree->selectionModel()->selectedRows().at(i));
        vec->push_front(info.fileName());
        vec->push_front(info.absoluteFilePath());
    }
    } else {
        vec->clear();
        *empty = true;
    }
}

////////// OTHER CODE //////////
CustomTree::CustomTree(QObject *parent) : QObject(parent)
{
    copy = new QAction(QString::fromUtf8("Копировать") ,this);
    cut = new QAction(QString::fromUtf8("Вырезать") ,this);
    paste = new QAction(QString::fromUtf8("Вставить") ,this);
    paste->setEnabled(false);
    erase = new QAction(QString::fromUtf8("Удалить") ,this);
    rename = new QAction(QString::fromUtf8("Переименовать"),this);
    mkdir = new QAction(QString::fromUtf8("Создать") ,this);
    prop = new QAction(QString::fromUtf8("Свойства") ,this);

    this->menu->setStyleSheet("border-style: outset;"
        "border-width: 1px;"
        "border-radius: 5px;"
        "border-color: beige;"
        "min-width: 10em;"
        "padding: 2px;"
        "border-color: black;");
    this->menu->addAction(copy) ;
    this->menu->addAction(cut) ;
    this->menu->addAction(paste);
    this->menu->addAction(erase);
    this->menu->addAction(rename);
    this->menu->addAction(prop) ;
    this->menu->addAction(mkdir);

    QObject::connect(this->prop,SIGNAL(triggered(bool)),
        this,SLOT(popupProp()));

    QObject::connect(this->copy,SIGNAL(triggered(bool)),
        this,SLOT(popupCopy()));

    QObject::connect(this->cut,SIGNAL(triggered(bool)),
        this,SLOT(popupCut()));

    QObject::connect(this->rename,SIGNAL(triggered(bool)),
        this,SLOT(popupRename()));

    QObject::connect(this->erase,SIGNAL(triggered(bool)),
        this,SLOT(popupErase()));

    QObject::connect(this->paste,SIGNAL(triggered(bool)),
        this,SLOT(popupPaste()));

    QObject::connect(this->mkdir,SIGNAL(triggered(bool)),
        this,SLOT(popupMkdir()));
}

void CustomTree::drawPath()
{
    auto index = tree->currentIndex();

    auto index2 = index;
    QString str =
        tree->model()->sibling(index2.row(),0,index2).data().toString();
    index2 = index2.parent();

    while ( index2.parent().data().toString() != NULL )
    {
        str = index2.data().toString() + slash + str;
        index2 = index2.parent();
    }
    str.insert(0,tab->tabText(tab->currentIndex()));

    adressLine->setText(str);
    fileNameLine->setText(index.sibling(index.row(),0).data().toString());
}

```

```

}
void CustomTree::openItem()
{
    if ( adressLine->text() != NULL )
    {
        QString str = "file:///";
        str.append(adressLine->text());
        QDesktopServices::openUrl(
            QUrl(str,
                QUrl::TolerantMode));
    }
}
void CustomTree::popup()
{
    checkSelected();
    if ( itemsSelected != 0 )
    {
        if (!*empty) paste->setEnabled(true);
        else paste->setEnabled(false);

        this->copy->setEnabled(true);
        this->cut->setEnabled(true);
        this->rename->setEnabled(true);
        this->erase->setEnabled(true);
    } else {
        if (!*empty) paste->setEnabled(true);
        else paste->setEnabled(false);
        this->copy->setEnabled(false);
        this->cut->setEnabled(false);
        this->rename->setEnabled(false);
        this->erase->setEnabled(false);
    }
    if (this->tree->currentIndex().isValid()
        &&
        !model->isDir(this->tree->currentIndex()))
        this->mkdir->setEnabled(false);
    else this->mkdir->setEnabled(true);
    menu->popup(QCursor::pos());
}

void CustomTree::popupErase()
{
    checkSelected();
    if ( itemsSelected != 0 )
    {
        popupCopy();
        auto it = vec->begin();
        qDebug() << vec->begin().operator *();
        while (it != vec->end())
        {
            //auto index = this->tree->selectionModel()->selectedRows(0).at(0);
            //qDebug() << index.data().toString();
            //bool ok = this->model->remove(index);
            boost::system::error_code error_code;

            QString file = it.operator *();
            // if ( boost::filesystem::is_directory(file.toStdString(),error_code) )
            //     boost::filesystem::remove_all(
            boost::filesystem::remove_all(file.toStdWString(),error_code );
            it += 2;

            if ( error_code )
            {
                QMessageBox::warning(this->tree, tr("Ошибка"),
                    tr("Ошибка удаления %1").arg(file)); continue;
            }
        }
        QString text = adressLine->text();
        while ( *(text.end() - 1) != slash )
        {
            text.remove(-1,text.length());
            //qDebug() << *text.end() << "\t" << text;
        }
        //text.remove(-1,text.length());
        adressLine->setText(text);
    }
}

void CustomTree::popupRename()
{

```

```

        checkSelected();
        auto index = this->tree->currentIndex();
        if ( itemsSelected != 0 ) this->tree->edit(
            tree->model()->sibling(index.row(),0,index));
    }
    void CustomTree::popupMkdir()
    {
        auto index = this->tree->currentIndex();
        if ( ! index.isValid() ) index = this->tree->rootIndex();

        QString name = QDialog::getText(tree, tr("Создание директории "),
            tr("Имя папки "));
        if ( !name.isEmpty() )
        {
            //qDebug() << adressLine << " " << name;
            QString newDir = adressLine->text() + slash + name ;
            if ( !boost::filesystem::create_directory(newDir.toStdWString()) )
                QMessageBox::information(tree, tr("Ошибка создания папки..."),
                    tr("Ошибка во время создания директории"));
        }
    }
    void CustomTree::popupProp()
    {
        auto info = this->model->fileInfo(this->tree->currentIndex());
        QString name = info.fileName();
        name += " - ";
        name += QString::number(info.size()) ;
        name += tr(" байт");

        propUi = new Prop(name);
        propUi->setNameEdit(info.fileName());
        propUi->setSizeEdit(QString::number(info.size()));
        propUi->setPathEdit(info.absoluteFilePath());
        propUi->show();
    }
    void CustomTree::addTree(QTreeView *tree)
    {
        this->tree = tree;
        this->tree->setDragEnabled(true);
        this->tree->setAcceptDrops(true);
        this->tree->setEditTriggers(QAbstractItemView::NoEditTriggers);
        this->tree->setSelectionMode(QTreeView::ContiguousSelection);
        this->tree->setContextMenuPolicy(Qt::CustomContextMenu);
        this->tree->setColumnWidth(0,200);
        this->tree->setColumnWidth(1,70);
        this->tree->setColumnWidth(2,60);
        this->tree->setSortingEnabled(true);

        QObject::connect(this->tree,SIGNAL(clicked(QModelIndex)),
            this,SLOT(drawPath()));

        QObject::connect(this->tree,SIGNAL(doubleClicked(QModelIndex)),
            this,SLOT(openItem()));

        QObject::connect(this->tree,SIGNAL(customContextMenuRequested(QPoint)),
            this,SLOT(drawPath()));

        QObject::connect(this->tree,SIGNAL(customContextMenuRequested(QPoint)),
            this,SLOT(popUp() ));
    }

    void CustomTree::checkSelected()
    {
        itemsSelected = this->tree->selectionModel()->selectedRows().length();
    }

#ifdef PROP_H
#define PROP_H

#include <QWidget>

namespace Ui {
class Prop;
}

class Prop : public QWidget
{

```

**Prop.h**

Q\_OBJECT

```
public:
    explicit Prop(QString name = "Title", QWidget *parent = 0);
    ~Prop();
    void setNameEdit(QString name);
    void setSizeEdit(QString name);
    void setPathEdit(QString name);

private slots:
    void on_closeButton_clicked();
    void closeButtonPress();
    void closeButtonRelease();

private:
    Ui::Prop *ui;
};

#endif // PROP_H
```

## Prop.cpp

```
#include "prop.h"
#include "ui_prop.h"

Prop::Prop(QString name, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Prop)
{
    ui->setupUi(this);

    this->setWindowTitle(name);

    QObject::connect(this->ui->closeButton, SIGNAL(pressed()),
                     this, SLOT(closeButtonPress()));
    QObject::connect(this->ui->closeButton, SIGNAL(released()),
                     this, SLOT(closeButtonRelease()));
}

Prop::~Prop()
{
    delete ui;
}

void Prop::setNameEdit(QString name)
{
    this->ui->nameEdit->setText(name);
}

void Prop::setSizeEdit(QString name)
{
    this->ui->sizeEdit->setText(name);
}

void Prop::setPathEdit(QString name)
{
    this->ui->pathEdit->setText(name);
}

void Prop::closeButtonPress()
{
    this->ui->closeButton->setStyleSheet(
        "border-style: outset;"
        "border-width: 1px;"
        "border-radius: 5px;"
        "border-color: beige;"
        "min-width: 7em;"
        "padding: 1px;"
        "border-color: blue;");
}

void Prop::closeButtonRelease()
{
    this->ui->closeButton->setStyleSheet(
        "border-style: outset;"
        "border-width: 1px;"
        "border-radius: 5px;"
        "border-color: beige;"
        "min-width: 7em;"
        "padding: 1px;"
        "border-color: grey;");
}

void Prop::on_closeButton_clicked()
{
    this->close();
}
```

## ПРИЛОЖЕНИЕ В

### Протокол системы контроля версий

Название	Описание
<b>Commits on Nov 16, 2017</b>	
Branch #1	—
<b>Commits on Nov 27, 2017</b>	
Change #2	Подключена библиотека boost, добавлена кроссплатформенность (не протестировано)
Change #3	Добавлена деактивация вкладки если диск имеет объем записи 0 байт. Протестирована кроссплатформенность в windows 10. Исправлены баги.

# ПРИЛОЖЕНИЕ Г

## Руководство программиста

Руководство содержит следующие основные разделы:




- Перечень классов (Рисунок 1);
- Описание класса CustomTree (Рисунок 2);
- Описание класса MainWindow (Рисунок 3);
- Описание класса Prop (Рисунок 4).


### My Project

[Main Page](#) [Classes ▾](#) [Files ▾](#)

**Class List**

Here are the classes, structs, unions and interfaces with brief descriptions:

 CustomTree	Класс посредник
 MainWindow	MainWindow - Главный фрейм, который создается автоматически
 Prop	Prop (Properties) - Класс отображающий свойство файла. Если в кратце, то просто очень сырой и не нужный

Generated by  1.8.13

### Рисунок 1 – Титульная страница

#### CustomTree Class Reference

Класс посредник [More...](#)

```
#include <customtree.h>
```

Inheritance diagram for CustomTree:



#### Public Slots

void <b>checkSelected</b> ()	Метод который проверяет количество выделенных файлов в рабочей области.
void <b>drawPath</b> ()	отрисовка пути до выделенного файла в UI.
void <b>openItem</b> ()	openItem - Конструкция, открывающая файл по нажатию двойного щелчка.
void <b>popup</b> ()	
void <b>popupCopy</b> ()	Метод, где все выделенные файлы загружаются в QList<QString>. Также используется и для вырезания, и для удаления, но немного иначе. <a href="#">More...</a>
void <b>popupCut</b> ()	
void <b>popupPaste</b> ()	Метод вставки выделенных файлов/каталогов.
void <b>popupErase</b> ()	Метод удаления выделенных файлов/каталогов.

### Рисунок 2 – Описание класса CustomTree



## MainWindow Class Reference

**MainWindow** - Главный фрейм, который создаётся автоматически. [More...](#)

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



### Public Slots

bool **event** (QEvent \*event)

### Public Member Functions

**MainWindow** (QWidget \*parent=0)

### Public Attributes

bool **empty** = true

Clipboard на самом деле это не буфер обмена, но нечто похожее. Эдакий внутри-программный буфер. Можно было воспользоваться библиотекой QClipboard, но я посчитал это слишком для моего маленького проекта.

bool **cutted** = false

int **selectedCount**

QModelIndex **selectedIndex**

### Detailed Description

## Рисунок 3 – Описание класса MainWindow

## Prop Class Reference

[Public Member Functions](#) | [List of all members](#)

**Prop** (Properties) - Класс отображающий свойство файла. Если в кратце, то просто очень сырой и не нужный. [More...](#)

```
#include <prop.h>
```

Inheritance diagram for Prop:



### Public Member Functions

**Prop** (QString name="Title", QWidget \*parent=0)

void **setNameEdit** (QString name)

void **setSizeEdit** (QString name)

void **setPathEdit** (QString name)

### Detailed Description

**Prop** (Properties) - Класс отображающий свойство файла. Если в кратце, то просто очень сырой и не нужный.

The documentation for this class was generated from the following files:

- [prop.h](#)
- [prop.cpp](#)

Generated by [doxygen](#) 1.8.13

## Рисунок 4 – Описание класса Prop

## **ПРИЛОЖЕНИЕ Д**

### **Доклад**

#### **1.1 Доклад**

Файловый менеджер разрабатывается для управления файловой системой операционной системы.

#### **1.2 Основная часть**

Основные задачи, решаемые файловым менеджером:

- Обеспечение быстрой и комфортной работы с файловой системой.

Основные достоинства:

- 1) Бесплатная;
- 2) Комфорт и быстрота работы;
- 3) Возможность просмотра свойств файла/каталога;

Основные недостатки:

- 1) Нет горячих клавиш (например: Del – быстрое удаление);

#### **1.3 Заключение**

Результаты моей работы заключается в