Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

ФАЙЛОВЫЙ МЕНЕДЖЕР

БГУИР КР 1-40 02 01 311 ПЗ

Студент:                                                           Латфулин В. Р.

Руководитель: Байдун Д. Р.

Минск 2021

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ..............................................................................................................3 Обзор методов и алгоритмов решения поставленной задачи...............................4 Обоснование выбранных методов и алгоритмов..................................................9 Описание программы для программиста.............................................................11 Описание алгоритмов для решения задачи..........................................................13 Руководство пользователя....................................................................................18 ЗАКЛЮЧЕНИЕ.....................................................................................................20 ЛИТЕРАТУРА.......................................................................................................21 Приложение А (Листинг кода) .............................................................................22 Приложение Б (Диаграмма классов) ...................................................................23 Приложение В (Блок-схема алгоритма поиска)……………..............................24 Приложение Г (Диаграмма последовательностей приложения).......................25

**ВВЕДЕНИЕ**

C++ – компилируемый, статически типизированный язык программирования. Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование и обобщенное программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. **C++** широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе х86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие.

Язык ассемблера — машинно-ориентированный язык программирования низкого уровня. Представляет собой систему обозначений, используемую для представления в удобно читаемой форме программ, записанных в машинном коду. Его команды прямо соответствуют отдельным командам машины или их последовательностям. Является существенно платформо-зависимым: языки ассемблера для различных аппаратных платформ несовместимы, хотя могут быть в целом подобны.

Язык ассемблера позволяет программисту пользоваться алфавитными мнемоническими кодами операций, по своему усмотрению присваивать символические имена регистрам ЭВМ и памяти, а также задавать удобные для себя схемы адресации (например, индексную или косвенную).

1. **Обзор методов и алгоритмов поставленной задачи**

Для создания этого программного обеспечения был выбрана технология Qt Creator. Qt Creator (ранее известная под кодовым названием Greenhouse) – кроссплатформенная свободная IDE для разработки на С,C++, JavaScript и QML. Разработана Trolltech (Digia) для работы с фреймворком Qt. Включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML.

Qt creator позволяет подключать разные библиотеки для соответствующих целей. Для работы с файловой системы были выбраны следующие библиотеки: QDir, QFile и QFileSystemModel.

* QDir используется для манипулирования путями, доступа к информации о путях и файлах, а также для манипулирования базовой файловой системой. Его также можно использовать для доступа к системе ресурсов Qt.
* QFile - это устройство ввода-вывода для чтения и записи текста, двоичных файлов и [ресурсов](https://doc.qt.io/qt-5/resources.html).
* QFileSystemModel - класс обеспечивает доступ к локальной файловой системе, предоставляя функции для переименования и удаления файлов и каталогов, а также для создания новых каталогов. В простейшем случае его можно использовать с подходящим виджетом отображения в составе браузера или фильтра.

С их помощью можно создавать директории, удалять директории и файлы, переименовывать их, а также есть вспомогательные методы для работы с файловой системой. В данном разделе будет рассмотрены основные функциональные составляющие проектируемого ПО.

**класс Fileoperation**

Рассмотрим основной элемент данного ПО: Системы обновления информации. Данный класс занимается обработкой запросов и обновлением информацией. Предназначение полей класса:

QString folderPath – переменная, которая хранит в себе адрес папки, в которой сейчас находится пользователь.

QString chosenPath – переменная, которая хранит себе адрес папки, которую выбрал пользователь одним нажатием мыши.

Методы класса:

bool Search(QString userDirName, QFileInfoList \*fileInfoList, QDir &cDir) – метод поиска папки с названием userDirName. Метод начинает поиск с папки cDir и проходит по всем внутренним папкам. Если он находит папку, название которой совпадает с параметром userDirName, то папка добавляется в список. Метод возвращает false в случае неудачного поиска. Метод является простым в реализации поставленной задачи.

bool CreateNewFolder(QString folderName) – метод, который создает папка с именем folderName в папке, в которой находится пользователь в данный момент. Возвращает false при существовании папки с таким же названием как и folderName. Достоинством данного метода является его простота решения поставленной задачи, простое написание программного. Недостатков по ходу работы не наблюдалось.

bool RenameFolder(QString newName) – метод, который переименовывает папку на новое название newName. В случае неудачи возвращает false. Метод является достаточно простое решение поставленной задачи, простое написание кода. Недостатков не наблюдалось.

void ContentDifference(QDir &sdir, QDir &dDir, QFileInfoList &diffList) - метод, который ищет соответствующие или неактуальные элементы в резервной папке. Сравнивает файлы папок sDir и dDir по последней дате редактирования. Использует метод recursiveContentList(QDir &dir, QFileInfoList &contentList). Данный метод решает поставленную задачу оптимальный способом.

void recursiveContentList(QDir &dir, QFileInfoList &contentList) - метод,

который наполняет список contentList всеми файлами и папками папки dir. Метод использует цикл и рекурсию вместе. Для поставленной задачи этот метод реализует одно из оптимальных решений поставленной задачи.

Гет и сет методы:

void SetFolderPath(QString folderPath) – сет-метод, который присваивает в приватное поле folderPath параметр функции.

void SetChosenPath(QString chosenPath) – сет-метод, который присваивает в приватное поле chosenPath параметр функции.

QString GetFolderPath() – гет-метод, который возвращает значение приватного поля foldetPath.

QString GetChosenPath() – гет-метод, который возвращает значение приватного поля chosenPath.

**класс** **Mainwindow** Данный класс является основных в работе с пользовательским окном. Он реализован с целью взаимодействия конечного пользователя данного программного продукта с системой обработки запросов и обновления данных. Данный класс содержит данные поля:

QFileSystemModel \*model – поле, с помощью которого производится работа с файловой системой.

FileOperation \*operation – поле, с помощью которого вызываются методы класса FileOperation. Button \*button – поле, c помощью которого осуществляется быстрый доступ к избранным папкам с помощью боковым кнопок. Также есть поля типа QShortcut для горячих клавиш: keyCmdB, keyCmdR, keyCmdD, keyCmdF. Каждое сочетание клавиш вызывает методы класса FileOperation.

Методы класса:

MainWindow() – конструктор класса, который инициализирует и соединяет кнопки быстрого перехода к одному методу, который выполняет одну работу для любой такой кнопки. Также метод инициализирует сочетания клавиш для основных операций и подключает их к соответствующим методам. Инициализируется окно и корневой путь.

~MainWindow() – деструктор класса, который удаляет окно и все используемые указателя на объекты классов. Это не обязательно, но освобождение памяти программистом после завершения программы так же считается хорошим тоном.

void on\_Search\_clicked() – метод, который срабатывает при нажатии кнопки поиска. Функция считывает пользовательскую строку и передает ее в метод Search класса FileOperation. Помимо этого в функции передается еще пустой список fileInfoList и папка cDir, с которой начинается поиск. После завершения метода Search метод on\_Search\_clicked обрабатывает возвращаемое значение и выполняет соответствующую работу. Недостатком данного метода является отображение первой записи списка. Метод простой в реализации поставленной задачи и маленький по объему программного кода.

void on\_Desktop\_clicked() – метод, которые позволяет получить быстрый доступ к избранным папкам по названию нажатой кнопки. Вызывает метод QString FastAccess(const QString buttonName) класса Button с помощью указателя на объект. Достоинством является простота решения поставленной задачи, простое написание кода.

void on\_actionBackup\_triggered() – метод, который реализовывает резервное копирование из одной папки в другую. У пользователя спрашивается разрешение на проведения метода. Если пользователь согласен, то происходит сравнение папок. Если папки одинаковые, то метод сообщит об ошибке и метод завершит работу. Сам метод вызывает метод contentDifference класса FileOperation, который добавляет в буферный список типа QFileInfoList. Далее происходит работа с содержимым этого списка. Данный метод решает поставленную задачу оптимальный способом. Одним недостатком является высокая вычислительная сложность алгоритма. Других недостатков по ходу работы не наблюдалось.

void on\_LvSource\_doubleClicked(const QModelIndex &index) – метод, который отслеживает двойное нажатие мыши пользователя по области. Сохраняет путь выбранной папки в поле FolderPath класса FileOperation. Осуществляет переход между папками в зависимости от выбора пользователя. Также присутствует возможность открытия исполняемых и обычных файлов. Достоинствами данного метода является его простота решения поставленной задачи, простой программный код. Одним недостатком является случай - при переходе в корневую папку пользователь получает доступ ко всем папкам устройства, тем самым может работать и повредить необходимые системные папки. Данный недостаток будет устранен группой программистов в ближайшем будущем.

void CreateFolder(const QString &fileName) – метод создает экземпляр класса Dialog для пользовательского ввода имени новой папки. Далее он соединяется с методом CreateNewFolder(QString) класса FileOperation с помощью указателя на объект. Метод вернет false при попытке создать папку с пустым или уже существующем именем. В таком случае будет показано окно с ошибкой. Метод является простым в исполнении и в написании кода.

void RenameFolder(const QString newFileName) – метод создает экземпляр класса Dialog для пользовательского ввода нового имени выбранной папки. Далее он соединяется с методом RenameFolder(QString) класса FileOperation с помощью указателя на объект. Метод вернет false при неудачной попытке переименовать папку. В таком случае будет показано окно с ошибкой. Метод является простым в исполнении и в написании кода.

void RemoveFolder() – метод, который сначала вызывает окно соглашения на удаление папки. Если пользователь действительно хочет удалить данную папку, то метод вызывает метод библиотеки QDir removeRecursive(), который удаляет каталог, включая все его содержимое. Если файл или каталог не могут быть удалены, алгоритм удаления продолжает работать и пытается удалить как можно больше файлов и подкаталогов, а затем возвращает false. Если же пользователь не согласен с удаление выбранной папки, то метод завершается.

Некоторые методы этого класса (для самой простой реализации данного приложения) избыточны, но были выбраны для удобного пользования программой и для расширения взаимодействия пользователя с программой.

класс **Dialog**

Данный класс реализован для получения информации от пользователя.

Методы класса:

Dialog() – конструктор, который инициализирует окно для пользовательского ввода. Также конструктор связывает кнопки с соответствующими методами.

void OkClicked() – метод, который вызывается при нажатии кнопки Ok окна Dialog.ui. Данный метод считывает строку, введенную пользователем. Далее метод связывается с соответствующим методом из класса MainWindow и возвращает строку, введенную пользователем. Метод является простым в исполнении и в написании кода.

Класс **Button**

Данный класс реализован для быстрого доступа к избранным папкам.

Методы класса:

QString FastAccess(QString &buttonText) – метод, который по названию кнопки возвращает соответствующий путь папки в метод on\_Desktop\_clicked() класса MainWindow. Метод является простым в исполнении и в написании кода.

Сочетания клавиш:

В разработанном программном обеспечении есть возомжность использовать сочетания клавиш для ускорения работы. Каждом сочетанию клавиш соответствует определенный метод.

Cmd + F(Ctrl + F в операционной системе Windows) – данное сочетание клавиш вызывает метод CreateFolder() класса MainWindow.

Cmd + В(Ctrl + D в операционной системе Windows) – данное сочетание клавиш вызывает метод on\_actionBackup\_triggered() класса MainWindow.

Cmd + D(Ctrl + D в операционной системе Windows) – данное сочетание клавиш вызывает метод RemoveFolder() класса MainWindow.

Cmd + R(Ctrl + R в операционной системе Windows) – данное сочетание клавиш вызывает метод RenameFolder() класса MainWindow.

Сама главная функция main() представляет собой простой цикл очистки и отображения в окно объектов, пока окно открыто. Это было целью, которая была успешно выполнена.

Большая часть методов этого программного обеспечения была разработана с помощью источников, которые приведены в литературе под пунктами 3-7.

1. **Обоснование выбранных методов и алгоритмов**

Метод on\_LvSource\_doubleClicked(const QModelIndex &index) класса MainWindow является одним из основых методов этого программного обеспечения. Именно он отображает все папки и файлы устройства пользователя и дает возможность перехода между папками. Также с помощью этого метода можно открывать текстовые и исполняемые файлы.

Данный алгоритм был выбран из-за своей простоты в реализации программного кода, небольшого объема и низкой вычислительной сложности алгоритма. Данный алгоритм не имеет аналогов. Простота реализации достигнута путем работы с библиотекой QDir и параметром, который хранит информацию о выбранном файле или папке. Точность метода достаточно высокая за счет получения полной информации о выбранном файле и документе, а алгоритму остается лишь только проверять тип файла и выполнять соответствующую операцию. Недостатком, а одновременно и особенностью, данного метода является то, что при переходе в корневую папку пользователь получает доступ ко всем папкам устройства и возможность редактирования их. Данный метод схож по работе с работой словаря, где по ключу можно получить значение. Вычислительная сложность алгоритма – O(n), где n – разновидности работы с файлами.

Алгоритм удаления. Этот алгоритм является одним из основополагающих алгоритмов этого программного обеспечения. Данный алгоритм написан с помощью библиотеки QDir, которая помогает разработчикам с разработкой собственных алгоритмов. Данный алгоритм прост в реализации программного кода, маленький по объему, но имеет высокую вычислительную сложность. Такая сложность достигнута путем прохода по всем составляющим удаляемой папки с помощью рекурсии. Основная работа выполнена с помощью внутренних методов библиотеки QDir. Точность данного алгоритма достаточно высокая, но есть вероятность, что метод не сможет удалить какие-то файлы в удаляемой папке. Вычислительная сложность алгоритма – O(), где n – количество внутренних папок.

Алгоритм создания новой папки. Алгоритм создает новую папку, название которой задает пользователь в отдельном окне. Алгоритм создает временную переменную типа QDir по указанному пути, который сохранен в классе FileOperation. Алгоритм работает в паре с классом Dialog.

Данный алгоритм простой в реализации поставленной задачи и небольшой по размеру программного кода. Основная работа выполнена с помощью внутренних методов библиотеки QDir. Вычислительная сложность алгоритма – O(1).

Алгоритм резервного копирования – копирует все недостающие папки и обновляет устаревшие файлы. Адекватных аналогов для этого алгоритма не представляется. Алгоритм непростой в реализации поставленной задачи и объемный по размеру программного кода. Имеет высокую вычислительную сложность алгоритма за счет обхода каждой папки. Вычислительная сложность – O(n2), где n – количество внутренних папок.

Алгоритм переименования папки. Алгоритм переименовывает название папки на новое, которое задает пользователь в новом отдельном окне. Алгоритм работает в паре с классом Dialog. Простой в реализации поставленной задачи и небольшой по размеру программного кода. Основная работа выполнена с помощью внутренних методов библиотеки QDir. Вычислительная сложность алгоритма – O(1).

Все конструкторы классов служат для инициализации полей и соединения между собой кнопок и соответствующих методов. Все деструкторы классов служат для удаления ненужной информации. Все конструкторы и деструкторы простые в реализации поставленной задачи и небольшие по размеру программного кода.

Алгоритм быстрого доступа к избранным папкам. Данный алгоритм имеет метод, который работает по принципу словаря. Он возвращает путь папки по названию кнопки. Данный метод схож по работе с работой словаря, где по ключу можно получить значение. После отработки метода алгоритм по отображает содержимое выбранной папки. Алгоритм простой в реализации поставленной задачи и маленький по размеру программного кода. Имеет среднюю вычислительную реализацию - O(n), где n – количество кнопок быстрого доступа.

Алгоритм поиска папки – реализует поиск папки, название которой совпадает с параметром userDirName, и, в случае удачи, добавляет папку в список fileInfoList. Поиск начинается с папки cDir. Метод простой в реализации поставленной задачи, небольшой по объему программного кода, но имеет очень высокую вычислительную сложность алгоритма. Такая вычислительная сложность достигнута из-за обхода большого количества папок и обхода всех составляющих этих папок. Точность алгоритма достаточно высокая, т.к. поиск идет по полном сравнению имен папок.

Все алгоритмы и методы реализованы так, что их выполнение начинается только после подтверждения пользователя на выполнение данного метода или алгоритма. Случайно алгоритм или метод выполняется не будет.

1. **Описание программы для программистов**

По ходу работы придерживались правила паттерна MVC. MVC — это паттерн проектирования веб-приложений, который включает в себя несколько более мелких шаблонов. При использовании MVC на три отдельных компонента разделены модель данных приложения, пользовательский интерфейс и логика взаимодействия пользователя с системой, благодаря чему модификация одного из этих компонентов оказывает минимальное воздействие на остальные или не оказывает его вовсе.

Основная цель применения MVC состоит в разделении данных и бизнес-логикиот визуализации. При использовании MVC упрощается и сопровождение программного кода: внесение изменений во внешний вид, например, не отражаются на бизнес-логике, а изменения бизнес-логикине затрагивают визуализацию.

Концепция MVC разделяет данные, представление и обработку действий пользователя на компоненты:

* Модель / Model — предоставляет собой объектную модель некой предметной области, включает в себя данные и методы работы с этими данными, реагирует на запросы из контроллера, возвращая данные и/или изменяя своё состояние. При этом модель не содержит в себе информации о способах визуализации данных или форматах их представления, а также не взаимодействует с пользователем напрямую.
* Представление / View — отвечает за отображение информации (визуализацию). Одни и те же данные могут представляться различными способами и в различных форматах. Например, коллекцию объектов при помощи разных представлений можно представить на уровне пользовательского интерфейса как в табличном виде, так и списком; на уровне API можно экспортировать данные как в JSON, так в XML или XSLX.
* Контроллер / Controller — обеспечивает связь между пользователем и системой, использует модель и представление для реализации необходимой реакции на действия пользователя. Как правило, на уровне контроллера осуществляется фильтрация полученных данных и авторизация — проверяются права пользователя на выполнение действий или получение информации.

В данном программном обеспечение роль представления берет на себя файл MainWindow.ui из системы MainWindow. Он служит для визуализации данных, работы и получения команд от пользователя. Команды могут возникать за счет боковых кнопок быстрого доступа, двойного нажатия по выбранной папке или файлу и с помощью сочетания клавиш.

Полученные команды обрабатываются в система файлов MainWindow.cpp и MainWindow.h. Данная система берет на себя работу контроллера. Cистема получает команды от MainWindow.ui, обрабатывает их и вызывает соответствующие методы других классов. Самой логики приложения в данной системе минимально. Система служит для передачи информации представлении и вызов соответствующих метод из модели.

Роль модели играет следующие системы файлов: FileOperation, Dialog, Button. В системе FileOperation осуществляется главные алгоритмы, которые вызываются из контроллера. Сам класс FileOperation представляет в себе набор методов, способных обновлять данные приложения, копировать, удалять, создавать, искать и переименовывать. Система файлов Button предназначена для обработки команд для быстрого доступа избранных папок. Система файлов Dialog состоит из Dialog.ui, Dialog.cpp и Dialog.hpp.

Если вызываются методы, которые требуют вводить информацию от пользователя, то вызывается окно Dialog.ui. Эта система реализована для обработки вызовов методов с участием пользовательского ввода информации.

Рис 3.1. Структура файлов

1. **Описание алгоритмов решения задач**

В данном разделе будут рассмотрены алгоритмы, используемые в разработанном ПО.

**Алгоритм визуализации содержимого папки и открытие файлов и папок**

Первый и ключевой алгоритм файлового менеджера – алгоритм вывода всего содержимого в выбранной папке и открытия файлов. Данный алгоритм реализован в методе on\_LvSource\_doubleClicked класса MainWindow. Метод принимает ссылку на индекс (QModelIndex) выбранной папки с именем index. Опишем его по шагам.

Шаг 1. Создание указателя listView на тот список содержимого, который вызвал данный метод.

Шаг 2. Создание структуры данных fileInfo о выбранной строке по индексу, в которой может находиться файлы, папки и переход к родительским и корневым папкам.

Шаг 3. Инициализируем поле folderPath класса FileOperatin с помощью структуры.

Шаг 4. Если имя строки в структуре равен "..", то создается временная переменная типа QDir, которая равна родительской папке папки, в которой сейчас находится пользователь. Далее идет отображение родительской папки с помощью указателя listView и переход к шагу 9. В противном случае – переход к шагу 5.

Шаг 5. Если имя строки в структуре равен ".", то идет отображение корневой папки устройства с помощью указателя listView и переход к шагу 9. В противном случае – переход к шагу 6.

Шаг 6. Если выбранная строка является папкой, то с помощью указателя listView отображаем все содержимое папки и переходим к шагу 9. В противном случае – переход к шагу 7.

Шаг 7. Если выбранная строка является исполняемым файлом, то создается переменная типа QProcess. Далее с помощью это переменной запускается выбранный исполняемый файл, который не привязан к разработанному программному обеспечению. Если исполняемый файл открыт, то осуществляется переход к шагу 9. В противном случае – переход к шагу 8.

Шаг 8. Если выбранная строка является любым файлом, но не исполняемым, то с помощью статического метода openUrl класса QDesktopServices открывается выбранный файл. Открытый файл не привязан к разработанному программному обеспечению. Если файл открыт, то осуществляется переход в шагу 9. В противном случае – переход к шагу 9.

Шаг 9. Завершение алгоритм.

**Алгоритм быстрого доступа к избранным папкам**

Данный алгоритм разработан в методе on\_Desktop\_clicked класса MainWindow. Опишем его по шагам.

Шаг 1. Создаем переменную textButton типа QString, в которую записываем название кнопки, которая вызвала данный метод.

Шаг 2. Вызываем метод FastAccess класса Button, который по названию кнопки определяет и возвращает путь избранной папки. Потом этот путь записывается в поле folderPath класса FileOperation.

Шаг 3. Через указатель на лист содержимого папки выводится содержимое избранной папки.

Шаг 4. Завершение алгоритма.

**Алгоритм резервного копирования**

Данный алгоритм разработан в методе on\_actionBackup\_triggered класса MainWindow. Данный алгоритм ничего не принимает и ничего не возращает. Опишем его по шагам.

Шаг 1. Создаем две переменных с именами sDir и dDir типа QDir, которые инициализируются папками из которой происходит копирование и в какую копируются данные соответственно.

Шаг 2. Если папки разные, то осуществляется переход к шагу 3. В противном случае – переход к шагу 14.

Шаг 3. Вывод окна соглашения на проведения данного алгоритма.

Шаг 4. Если пользователь согласен, то осуществляется переход к шагу 5. В противном случае – переход к шагу 14.

Шаг 5. Создание списка diffList структур информации о папках и файлах.

Шаг 6. Вызов метода contentDifference, в который передаются sDir, dDir и diffList. Данный алгоритм заполняет список diffList неактуальными файлами и недостающими папками.

Шаг 7. Запускается цикл foreach, в котором создается переменная diffInfo типа fileInfo для обхода по спику diffList. Цикл работает до тех пор, пока не произойдет обход всех папок или файлов списка diffList.

Шаг 8. Создание переменной backup типа QString. Переменная backup инициализируется путем diffInfo, который заменяется на путь папки dDir.

Шаг 9. Если структура данных diffInfo содержит файл – переход к шагу 10. Если содержит папку – переход к шагу 12.

Шаг 10. Происходит удаление файла папки dDir.

Шаг 11. Происходит копирование файла из папки sDir в папку dDir. Переход к шагу 13.

Шаг 12. Происходит копирование всей папки структуры diffInfo в папку dDir. Переход к шагу 13.

Шаг 13. Запуск новой итерации по циклу foreach и переход к шагу 8.

Шаг 14. Завершение алгоритма.

**Алгоритм удаления выбранной папки**

Следующий алгоритм – удаление выбранной папки. Данный алгоритм описан в методе on\_actionRemove\_triggered класса MainWindow. Опишем данный метод с помощью блок-схемы(см. рисунок 2).

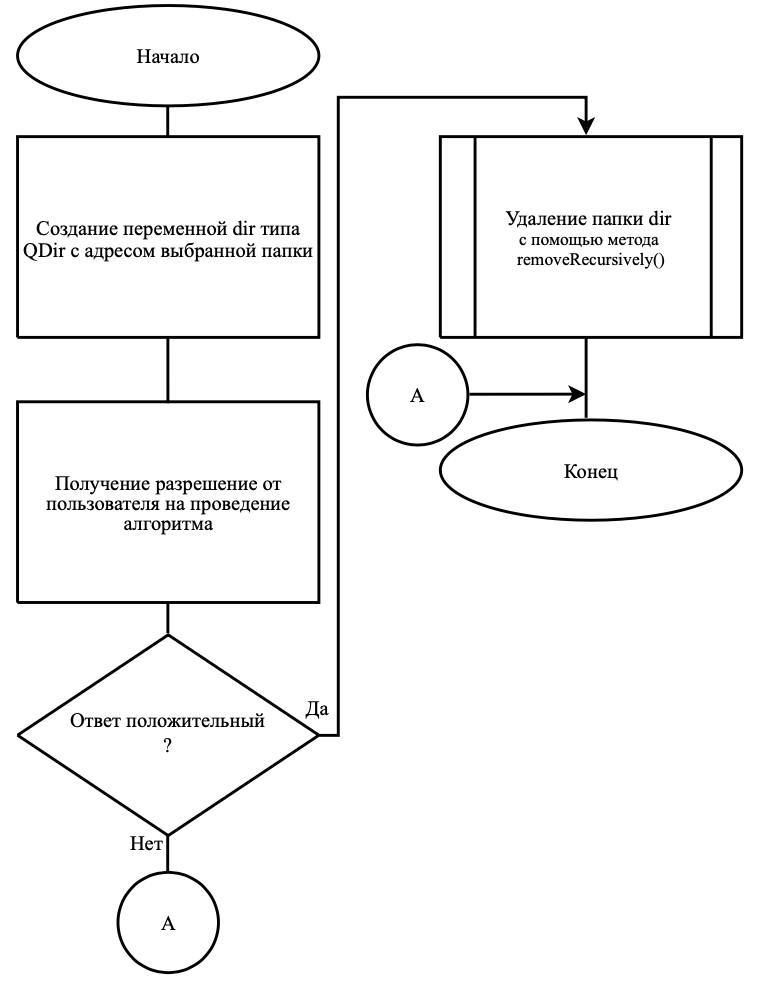


Рис. 4.1. Алгоритм удаления выбранной папки.

**Алгоритм создания новой папки**

Следующий алгоритм – создание новой папки. Данный алгоритм описан в методе on\_actionNew\_Folder\_triggered() класса MainWindow. Данный метод работает в паре с классом Dialog. Опишем данный метод с помощью блок-схемы(см. рисунок 3).

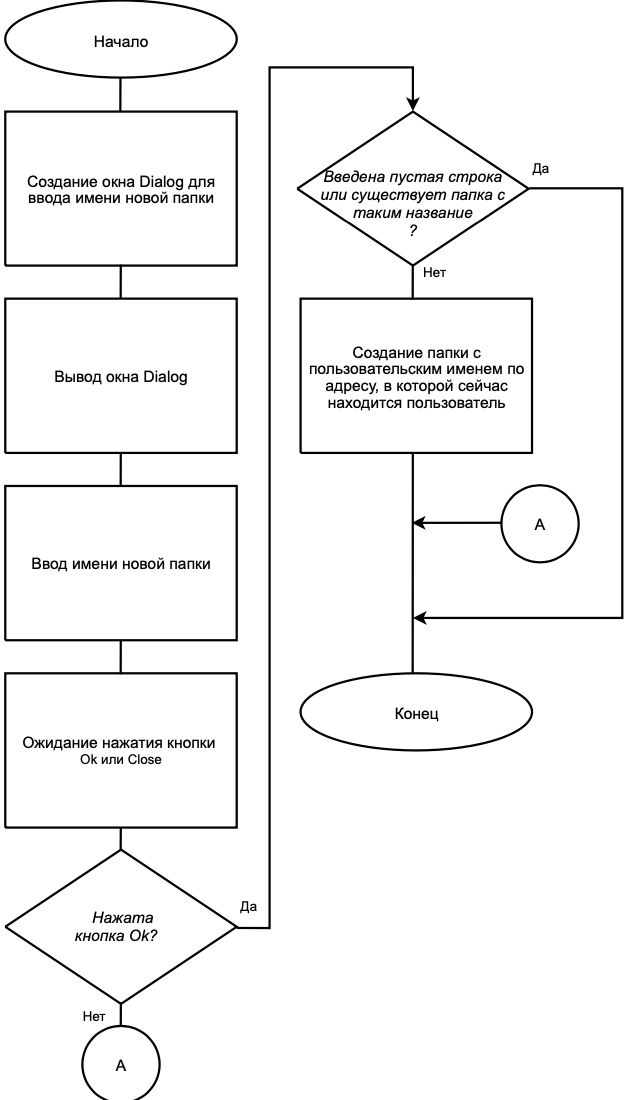


Рис. 4.2. Алгоритм создания новой папки.

**Алгоритм поиска папок с определенным названием**

Данный алгоритм описан в методе Search класса FileOperation. Метод принимает ссылку на папку, с которой осуществляется поиск, ссылку на список структур, которые запоминают всю информацию о папке или о файле, и название ищущей папки. Данный метод описан с помощью блок-схемы в приложение Б.

**Алгоритм получения всех файлов и папок папки**

Данный алгоритм используется в алгоритме резервного копирования. Данный алгоритм описан в методе recursiveContentList класса FileOperation. Данный метод принимает ссылку на папку(&QDir) с именем dir , все файлы и папки будут получены, и ссылку на список структур(QFileInfoList) с именем infoList, который предназначен для хранения информации о файле или папке. Опишем данный алгоритм по шагам.

Шаг 1. Запускается цикл foreach, в котором создается переменная info типа fileInfo. В эту переменную по очереди будут записываться вся информация о файле или о папке. Цикл будет длиться, пока не обойдет весь список файлов и папок папки dir.

Шаг 2. Добавление структуры info в список infoList.

Шаг 3. Если структура info содержит информацию о папке, то осуществляется переход в эту папку и переход к шагу 4. В противном случае – переход к шагу 6.

Шаг 4. Рекурсивный вызов метода recursiveContentList, в которую мы передаем обновленную переменную dir и список структур infoList.

Шаг 5. Переход в родительскую папку dir.

Шаг 6. Запуск новой итерации по циклу foreach и переход к шагу 2.

Шаг 7. Завершение алгоритма.

Данные алгоритмы являются достаточными, для организации простого ввода, обновления и вывода данных. Благодаря алгоритмам создания, удаления, резервного копирования можно работать с логической частью программы.Так же стоит заметить, что благодаря организации взаимодействия с кнопками пользователю стали доступны и другие функции, которые не обязательны, но созданы для упрощённого пользования программой. Все эти алгоритмы были созданы для простого использования программы и понятной главной функции.

1. **Руководство пользователя**

В данном разделе приведена вся информация, которая должна публиковаться вместе с продуктом. В данной программе для корректного функционирования требуются следующие файлы: Папки: cmake-build-release, для, непосредственно, запуска приложения в .exe формате (при желании можно запустить main.cpp и скомпилировать), include для загрузки заголовочных файлов, src для загрузки .cpp файлов. Файлы: MakeFile для упрощённой сборки приложения.

Далее будет представлены снимки экранов из приложения с пояснением каждых функций.

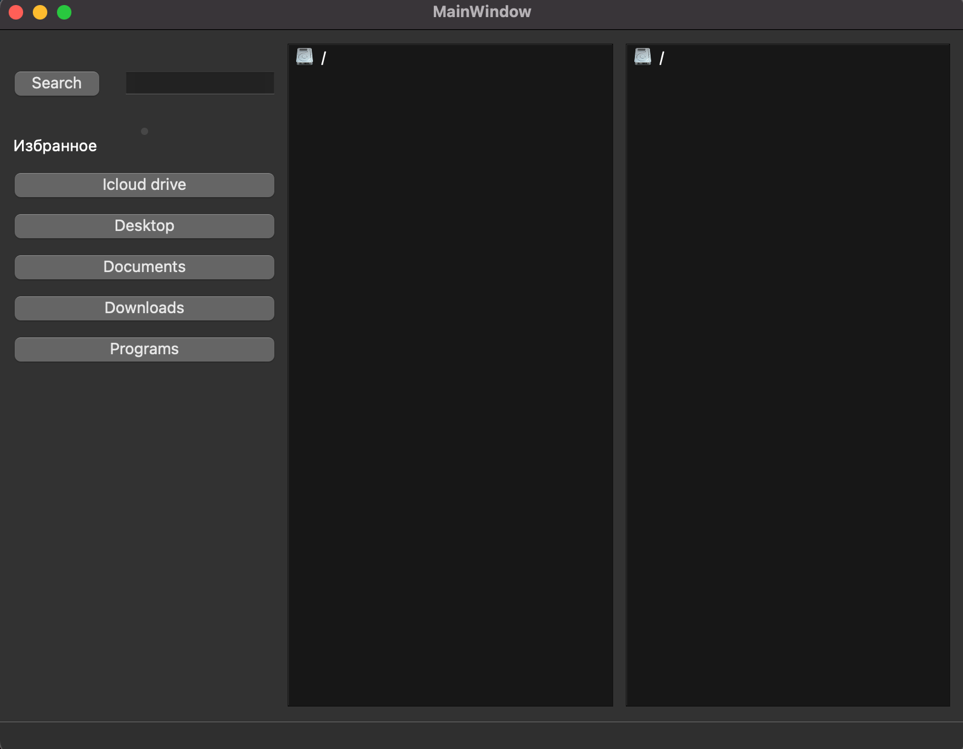


Рисунок 5.1 – основные поля для взаимодействия и вывода информации

В данном рисунке показано два поля для взаимодействия и вывода информации. Для взаимодействия с полем необходимо произвести двойное нажатие мыши. Слева от полей взаимодействия есть кнопки быстрого доступа, благодаря которым можно быстро получить нужную пользователю папку из списка избранных. Если нужной папки нет в данном списке, то можно обратиться к поиску, который расположен над кнопками.

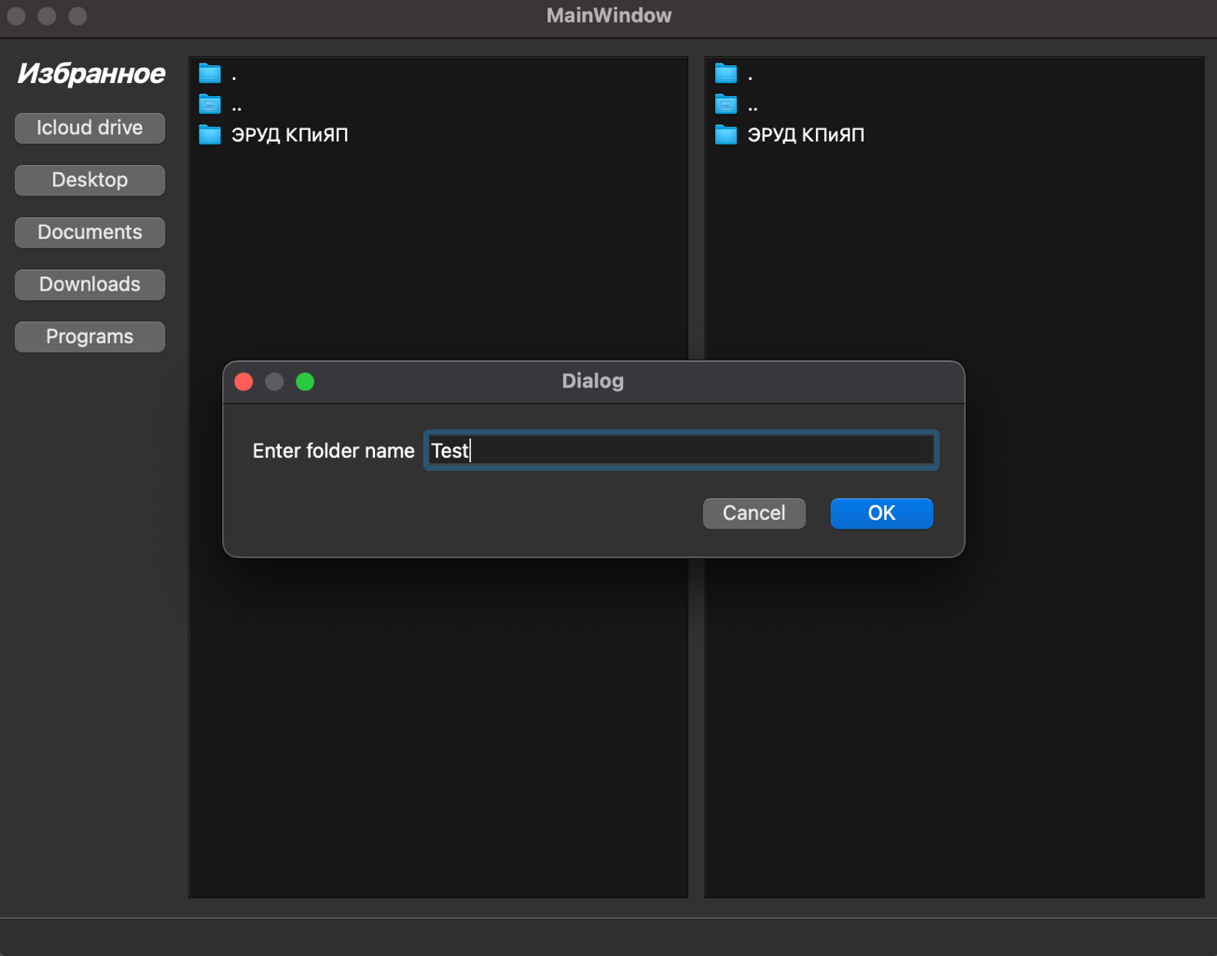


Рисунок 5.2 – диалоговое окно для ввода информации от пользователя

На рисунке 5.2 показано диалоговое окно, которое служит для ввода информации от пользователя. Данное окно появляется в двух случаях: при создании новой папки, имя которой вводит пользователь, и переименование имени папки на новое.

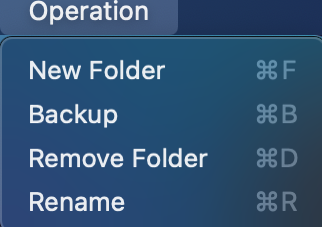


Рисунок 5.3 – основное меню операции над папками.

На рисунке 5.3 показано основное меню операций, который пользователь может совершить над папками. Справа от названий операций написаны сочетания клавиш, с помощью которых выполнится соответствующая операция

**Заключение**

В данном разделе будут проведены итоги разработки программного обеспечения.

В ходе разработки и тестирования данного ПО было проверено и практически обосновано, что оно выполняет все требуемые функции:

* Переход между папками;
* Открытие файлов;
* Операции над папками;

Для этого была использованы библиотеки по работе с файловой системой, которая помогала в реализации собственных алгоритмов.

По ходу разработки данного программного обеспечения были получены знания по работе с объектно-ориентированным программировании, с библиотеками с файловой системой, с проектированием программного обеспечения, с пользовательским интерфейсом.

В данном программном обеспечении были реализованы основные операции над папками: удаление, переименование, копирование, создание.

Все эти операции можно вызвать с помощью сочетания клавиш.

В дальнейшем данный программный продукт может быть развит с помощью добавления новых функций над папками, которые будут удобны и востребованы пользователю и расширения функционала мыши в этой программе.

Таким образом, данный программный продукт получил хорошее развитие в процессе написания данного курсового проекта, но также может разрабатываться дальше с целью получения коммерческой выгоды, например.

Данный курсовой проект помог понять, как устроены многие приложения, показать их недостатки и понять, как их можно усовершенствовать.

**Литература**

1. Лафоре Р. Объектно-ориентированное программирование с С++/ 4-е издание М.:Питер, 2004. – 923 с.
2. Луцик Ю.А. Объектно-ориентированное программирование на языке С++ / Ю.А.Луцик, А.М.Ковальчук, И.В.Лукьянова. – Мн.: БГУИР, 2003
3. Документация библиотеки QDir. – [Электронный ресурс]. – Адрес ресурса: <https://doc.qt.io/qt-5/qdir.html> - Дата доступа 08.12.2021
4. Документация библиотеки QFileInfo – [Электронный ресурс]. – Адрес ресурса: <https://doc.qt.io/qt-5/qfileinfo.html#isSymLink> - Дата доступа 08.12.2021
5. Документация библиотеки QFileSystemmodel – [Электронный ресурс]. – Адрес ресурса: <https://doc.qt.io/qt-5/qfilesystemmodel.html> - Дата доступа 08.12.2021
6. Документация библиотеки QProcess. – [Электронный ресурс]. – Адрес ресурса: <https://doc.qt.io/qt-5/qprocess.html> - Дата доступа 08.12.2021
7. Документация библиотеки Qshortcut. – [Электронный ресурс]. – Адрес ресурса: <https://doc.qt.io/qt-5/qshortcut.html> - Дата доступа 08.12.2021
8. Применение паттерна проектирования MVC. – [Электронный ресурс]. – Адрес ресурса: <https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami> - Дата доступа 08.12.2021

**ПРИЛОЖЕНИЕ А**

(Обязательное)

Листинг кода

**ПРИЛОЖЕНИЕ Б**

(Обязательное)

Диаграмма классов приложения

**ПРИЛОЖЕНИЕ В**

(Обязательное)

Блок-схема алгоритма поиска

**ПРИЛОЖЕНИЕ Г**

(Обязательное)

Диаграмма последовательностей приложения