Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ
И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

ДВУХПАНЕЛЬНЫЙ ФАЙЛОВЫЙ МЕНЕДЖЕР С ВКЛАДКАМИ

БГУИР КП 1-40 02 01 308 ПЗ

Студент: группы 150503,
Каленчиц А.В.

Руководитель: ассистент кафедры ЭВМ
Басак Д.В.

МИНСК 2023

СОДЕРЖАНИЕ

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Задание на курсовой проект – двухпанельный файловый менеджер с вкладками. Файловый менеджер должен выполнять наиболее частые операции с файлами:

- создание;
- открытие/проигрывание/просмотр;
- редактирование;
- перемещение;
- переименование;
- копирование;
- удаление;
- поиск;
- гибкий запуск программ для работы с файлами.

Также необходимо, чтобы файловый менеджер позволял выполнять основные операции с вкладками:

- создание;
- закрытие;
- перемещение по вкладкам.

ВВЕДЕНИЕ

Программы бывают разными, большими и маленькими, популярными и малоизвестными, редко используемыми и постоянно запущенными… К числу часто используемых и широко распространенных программ относятся файловые менеджеры.

В первых ОС практически отсутствовал интерфейс пользователя. Была только командная строка, в которую пользователь вводил команды, набирая их на клавиатуре, что вызывало большие неудобства. Позднее появились программы, визуально отображающие содержание дисков, позволившие нажатием одной клавиши запускать программы, и выполнять различные файловые операции. Эти программы стали называть «файловыми оболочками», так как они выполняли функцию посредника между операционной системой и пользователем. С появлением графического интерфейса пользователя в операционных системах программы-оболочки потеряли свое значение и стали называться «диспетчеры файлов» или «файловые менеджеры».

Названий у теперь уже файловых менеджеров было много: надстройки над операционной системой, файловые оболочки, файловые менеджеры, а попросту - «коммандеры». Они появились почти одновременно с ПК. Их породили ДОС - дисковые операционные системы, чьи многочисленные команды вводились с клавиатуры и отличались таким разнообразием форматов, что казались любому непрограммисту китайской грамотой. Так и возникла все еще актуальная проблема интерфейса для человека с минимальным уровнем подготовки.

Что же должен «делать» такой интерфейс? Быть посредником между пользователем и операционной системой. Максимально упрощать создание, упорядочение, копирование, перенос и поиск материалов. В идеале такой интерфейс должен быть понятен сразу, без изучения.

При разработке программы учитывается множество факторов. Конечный продукт должен обладать достаточным количеством возможностей, чтобы заинтересовать пользователя, но при этом иметь простой, доступный интерфейс и приемлемую скорость. Например, файловый менеджер AccelMan, безусловно, заслуживает самых лестных слов в плане функциональности. Но при этом разработчики забывают о том, что файловый менеджер - это один из самых первых инструментов, устанавливаемых на рабочей станции. Соответственно, программа должна работать максимально быстро и не забивать голову пользователя необходимостью изучения десятков кнопок и пунктов меню.

Наверное, не было такого этапа в истории ПК, который не породил бы программу, с большим или меньшим успехом выполняющую перечисленные функции. Но, бесспорно, истинный долгожитель в этом смысле всем известный Norton Commander. Его разработал американский программист

Питер Нортон. Файловый менеджер наглядно показывал на экране всю файловую структуру компьютера: диски, каталоги и файлы. С такой программой не надо было набирать сложные команды MS-DOS в командной строке. Само слово «Нортон» в начале 90-х годов воспринималось как обозначение любого файлового менеджера. Впрочем, немного позже среди русскоязычных пользователей заслуженную популярность завоевал Dos Navigator, обладающий массой новых, по тем временам уникальных возможностей.

Total Commander нельзя назвать самой функциональной, самой красивой или самой быстрой программой данного класса. Причина популярности файлового менеджера кроется в сбалансированности всех составляющих. Один из первых файловых менеджеров, Norton Commander, оказался столь удачным, что породил целый класс аналогичных программ, названных Orthodox File Managers (OFMs) - «классические файловые менеджеры». С развитием компьютерной техники их функции совершенствовались, но главная идея оставалась неизменной. Windows Commander, Volkov Commander, DOS Navigator, FAR и другие популярные программы заимствовали «джентльменский набор» NC и дополнили его новыми возможностями : графическим интерфейсом, средствами работы в сети и даже играми.

Однако уже довольно давно возник логичный вопрос: не пришло ли время для чего-то совершенно нового? Вначале, когда NC был чуть ли не единственным средством для работы с файлами, его командам волей-неволей должен был учиться всякий пользователь ПК. Сегодня о программах этого типа уже говорят как о предназначенных для опытных пользователей. Похоже, классические «коммандеры» освободили нишу для средств управления файлами, которые «общаются» с пользователем на более понятном ему языке. И эта ниша пока пустует...

1 ОБЗОР ЛИТЕРАТУРЫ

Файловый менеджер (англ. file mannager) — компьютерная программа, предоставляющая интерфейс пользователя для работы с файловой системой и файлами. Файловый менеджер позволяет выполнять наиболее частые операции:

- создание;
- открытие/проигрывание/просмотр;
- редактирование;
- перемещение;
- переименование;
- копирование;
- удаление;
- изменение атрибутов и свойств;
- поиск файлов;
- назначение прав;
- гибкий запуск программ для работы с файлами.

Помимо основных функций, многие файловые менеджеры включают ряд дополнительных возможностей, например, таких как работа с сетью (через FTP, NFS и т. п.), резервное копирование, управление принтерами и пр.

Современный файловый менеджер должен: обеспечивать удобную возможность работы с файлами, запускать внешние программы для работы с разными типами файлов, позволять с легкостью и удобством работать как клавиатурой, так и с помощью мышки, поддерживать технологию плагинов и настройку цветовых схем.

В состав базового дистрибутива файлового менеджера должны входить ряд модулей, плагинов, которые непосредственно связаны с работой с файлами:

1. Просмотр и редактирование текстовых файлов, подцветка синтаксиса, поддержка разных кодировок (включая Unicode).
2. Поиск и замена по множеству файлов, множественное переименование файлов, просмотр картинок, работа с архивами.

Выделяют три основных типа файловых менеджеров: навигационные, двухпанельные и консольные.

Навигационные (пространственные) файловые менеджеры. Представители данного типа при работе с очень большим количеством файлов значительно заменяют работу системы. Отсутствие второй панели для копирования или перемещения файлов иногда расценивается как недостаток данного типа файловых менеджеров.

Навигационные файловые менеджеры: проводник Windows (англ. Windows Explorer) — встроен в Windows, Mac OS X, Finder, Path Finder, POSIX (Linux, BSD и т.д.), Konqueror — поставляется с KDE, Nautilus (файловый менеджер) — поставляется с GNOME.

Двухпанельные (ортодоксальные) файловые менеджеры. Наиболее распространенный вид «коммандерров» из-за удобного интерфейса и гибкого функционала. В общем случае имеют две равноценные панели для списка файлов, дерева каталогов и т. п.

Наиболее известные ортодоксальные файловые менеджеры: Norton Commander, Dos Navigator, Volkov Commander, PIE Commander, FAR Manager, Total Commander, POSIX (Linux, BSD и т. д.), Midnight Commander, Krusader, GNOME Commander.

Консольные файловые менеджеры. Такие менеджеры могут быть очень полезны в повседневных задачах, при управлении файлами на локальном компьютере. Консольным файловым менеджерам требуется меньше системных ресурсов, чем аналогичным по функциональности файловым менеджерам с графическим интерфейсом. Консольными файловыми менеджерами являются FAR Manager, GNU Midnight Commander и др.

Таким образом, файловые менеджеры – специальные программы, предназначенные для облегчения общения пользователя с командами операционной системы. Это программы, запускаемые под управлением ОС, занимающие промежуточное положение между ОС и прикладными программами и служащие для интеграции прикладных пакетов.

В данном курсовом проекте будет представлен двухпанельный файловый менеджер.

Для написания программы и реализации графического интерфейса была выбрана среда разработки Qt Creator, так как она включает фреймворк Qt, который является полезным инструментом в реализации программ, отражающих работу внутренней системы устройства.

Qt — фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++, который является полностью объектно-ориентированным и позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения (элементы графического интерфейса, классы для работы с базами данных).

Язык программирования C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также компьютерных игр. Поэтому данный язык программирования является хорошим выбором для реализации курсового проекта.

2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

Диаграмма классов UML представлена в ПРИЛОЖЕНИИ В.

Для создания двухпанельного файлового менеджера с вкладками в программе были реализованы различные классы в зависимости от требуемого функционала.

2.1 Класс CreateChoice

Класс CreateChoice реализует работу всплывающего окна для выбора системного объекта при его создании.

Атрибуты класса:

- `Ui::CreateChoice *ui` - указатель для связи с соответствующим ui-файлом;
- `QString fileName` - переменная для хранения имени созданного файла;
- `QString dirName` - переменная для хранения имени созданного каталога;
- `QString linkName` - переменная для хранения имени символьной ссылки;
- `bool file` - флаг выбора файла;
- `bool dir` - флаг выбора каталога;
- `bool link` – флаг выбора символьной ссылки.

Методы класса:

- `bool get_file()` - метод, сообщающий о выборе создании файла;
- `bool get_dir()` - метод, сообщающий о выборе создания каталога;
- `bool get_link()` - метод, сообщающий о выборе создания каталога.

Слоты класса:

- `void on_btnFile_clicked()` - слот выбора файла;
- `void on_btnDir_clicked()` - слот выбора каталога;
- `void on_btnLink_clicked()` – слот выбора символьной ссылки;
- `void on_btnCancel_clicked()` - слот нажатия на кнопку «Cancel».

2.2 Класс Form

Данный класс отображает внешний вид и работу большинства элементов интерфейса основного окна приложения.

Атрибуты класса:

- `Ui::Form *ui` - указатель для связи с соответствующим ui-файлом;
- `QFileSystemModel *model` - указатель модели данных файловой системы;
- `QThread *threadCopy` – указатель на объект для копирования в отдельном потоке;
- `QThread *threadRemove` - указатель на объект для удаления в отдельном потоке;
- `QThread *threadReplace` - указатель на объект для перемещения в отдельном потоке;
- `QThread *threadSearch` - указатель на объект для поиска в отдельном потоке;
- `QListView *view` – указатель на объект класса QListView;
- `SearchResult *window` – указатель на объект класса SearchResult;
- `ThreadToCopy *thCopy` – указатель на объект потока копирования;
- `ThreadToRemove *thRemove` – указатель на объект потока удаления;
- `ThreadToReplace *thReplace` – указатель на объект потока перемещения;
- `ThreadToSearch *thSearch` – указатель на объект потока поиска;
- `File f` - объект класса File для выполнения операций с текстовыми файлами;
- `Dir d` - объект класса Dir для выполнения операций с каталогами;
- `SysElem *file` - указатель на объект класса SysElem;
- `SysElem *dir` - указатель на объект класса SysElem;
- `QString filePath` – переменная для хранения пути файла;
- `QString dirPath` - переменная для хранения пути каталога;
- `QString searchName` – переменная для хранения имени искомого системного объекта.

Методы класса (методы-обертки для слотов):

- `void btn_create();`
- `void btn_remove();`
- `void btn_copy();`
- `void btn_replace();`
- `void btn_rename();`
- `void btn_search().`

Слоты класса:

- `void on_lvLeft_clicked(const QModelIndex&)` - слот нажатия на панель `lvLeft`;
- `void on_lvLeft_doubleClicked(const QModelIndex&)` - слот двойного нажатия на панель `lvLeft`;
- `void on_btnCreate_clicked()` - слот нажатия на кнопку «Create»;
- `void on_btnRemove_clicked()` - слот нажатия на кнопку «Remove»;
- `void on_btnCopy_clicked()` - слот нажатия на кнопку «Copy»;
- `void on_btnReplace_clicked()` - слот нажатия на кнопку «Replace»;
- `void on_btnRename_clicked()` - слот нажатия на кнопку «Rename»;
- `void on_lineSearch_textEdited(const QString&)` - слот ввода имени для поиска;
- `void on_btnSearch_clicked()` - слот нажатия на кнопку «Search»;
- `void on_leftPath_textEdited(const QString&)` - слот ввода пути для отображения файлов на панели `lvLeft`;
- `void remove_is_not_performed()` – слот обработки сигнала о сбое удаления;
- `void copy_is_not_performed()` – слот обработки сигнала о сбое копирования;
- `void replace_is_not_performed()` – слот обработки сигнала о сбое перемещения;
- `void ready_to_remove()` – слот обработки сигнала о завершении удаления;
- `void ready_to_copy()` – слот обработки сигнала о завершении копирования;
- `void ready_to_replace()` – слот обработки сигнала о завершении перемещения;
- `void ready_to_search(QFileInfoList)` – слот обработки сигнала о завершении поиска.

Сигналы класса:

- `void start_copy(QDir, SysElem*, SysElem*, QString, QString)` – сигнал о выполнении копирования;
- `void start_remove(SysElem*, SysElem*, QString, QString)` – сигнал о выполнении удаления;
- `void start_replace(QDir, SysElem*, SysElem*, QString, QString)` – сигнал о выполнении перемещения;

- `void start_search(QString, QString, QString)` – сигнал о выполнении поиска.

2.3 Класс LinkedPath

Класс LinkedPath создан для записи пути связываемого с символьной ссылкой системного объекта во всплывающем окне.

Атрибуты класса:

- `Ui::LinkedPath *ui` - указатель для связи с соответствующим ui-файлом;
- `QString path` - переменная для хранения нового имени выбранного файла.

Методы класса:

- `QString get_path()` - метод передачи пути.

Слоты класса:

- `void on_btnCancel_clicked()` – слот нажатия на кнопку «Cancel»;
- `void on_btnOK_clicked()` - слот нажатия на кнопку «OK»;
- `void on_path_textEdited(const QString&)` – слот для ввода пути связываемого файла.

2.4 Класс MainWindow

Класс, связанный с интерфейсом основного окна приложения. Содержит функционал таких элементами интерфейса, как вкладки, информационное поле и т.д. В данном классе находится реализация взаимодействия с интерфейсом с помощью комбинаций клавиш.

Атрибуты класса:

- `Ui::MainWindow *ui` - указатель для связи с соответствующим ui-файлом;
- `Form *form` – указатель на объект класса Form.

Слоты класса:

- `void add_tab()` – слот для создания вкладки;
- `void on_tabWidget_tabCloseRequested(int)` – слот нажатия на виджет вкладки;
- `void on_CtrlX_triggered()` – слот срабатывания комбинации клавиш Ctrl + X;
- `void on_CtrlC_triggered()` – слот срабатывания комбинации клавиш Ctrl + C;
- `void on_CtrlEsc_triggered()` - слот срабатывания комбинации клавиш Ctrl + Esc;

- `void on_CtrlN_triggered()` - слот срабатывания комбинации клавиш Ctrl + N;
- `void on_CtrlDel_triggered()` - слот срабатывания комбинации клавиш Ctrl+Delete;
- `void on_CtrlT_triggered()` - слот срабатывания комбинации клавиш Ctrl + T;
- `void on_CtrlR_triggered()` - слот срабатывания комбинации клавиш Ctrl + R;
- `void on_CtrlD_triggered()` - слот срабатывания комбинации клавиш Ctrl + D;
- `void on_CtrlF_triggered()` - слот срабатывания комбинации клавиш Ctrl + F.

2.5 Класс NewName

Данный класс предназначен для возможности переименования системного объекта во всплывающем окне, а также для возможности создания имени нового системного объекта.

Атрибуты класса:

- `Ui::NewName *ui` - указатель для связи с соответствующим ui-файлом;
- `QString name` - переменная для хранения нового имени выбранного файла.

Методы класса:

- `QString get_name()` - метод, возвращающий новое имя системного объекта.

Слоты класса:

- `void on_name_textEdited(const QString&)` - слот ввода нового имени;
- `void on_btnOK_clicked()` - слот нажатия на кнопку «OK»;
- `void on_btnCancel_clicked()` - слот нажатия на кнопку «Cancel».

2.6 Класс SearchResult

Класс SearchResult реализует вывод результатов поиска системного объекта по имени, указанном в главном окне приложения в соответствующей текстовой строке. Также в нем представлена функция взаимодействия с буфером обмена для быстрого перехода к расположению результата поиска.

Атрибуты класса:

- `Ui::SearchResult *ui` - указатель для связи с соответствующим ui-файлом.

Методы класса:

- `void set_ui(QFileInfoList)` - метод передачи результатов поиска для отображения;
- `void reset_ui()` - метод очистки окна отображения результатов поиска.

Слоты класса:

- `void on_btnOK_clicked()` - слот нажатия на кнопку «OK»;
- `void on_leftList_itemClicked(QListWidgetItem*)` – слот нажатия на панель `leftList`.

2.7 Класс SysElem

Данный класс, предназначенный для работы с системными объектами, является абстрактным базовым классом для двух производных от него классов: File и Dir. Класс SysElem является ключевой частью управления системными объектами в данном приложении.

Методы класса:

- `bool create()` – чисто виртуальный метод создания системного объекта;
- `bool r_move()` – чисто виртуальный метод удаления системного объекта;
- `bool r_name(QString, QString)` – чисто виртуальный метод переименования системного объекта;
- `bool c_py(QString)` - чисто виртуальный метод копирования системного объекта.

2.8 Класс File

Класс File предназначен для работы с текстовыми файлами.

Методы класса:

- `bool create()` – метод создания системного объекта;
- `bool r_move()` – метод удаления системного объекта;
- `bool r_name(QString, QString)` – метод переименования системного объекта;
- `bool c_py(QString)` - метод копирования системного объекта.

2.9 Класс Dir

Класс Dir предназначен для работы с каталогами.

Методы класса:

- `bool create()` – метод создания системного объекта;

- `bool r_move()` – метод удаления системного объекта;
- `bool r_name(QString, QString)` – метод переименования системного объекта;
- `bool c_py(QString)` - метод копирования системного объекта.

2.10 Класс ThreadToCopy

Данный класс создан для выполнения копирования системных объектов в отдельном потоке.

Методы класса:

- `void c_py(QDir, SysElem*, SysElem*, QString)` – метод выполнения копирования.

Слоты класса:

- `void run_copy(QDir, SysElem*, SysElem*, QString, QString)` – слот-обертка для копирования в отдельном потоке.

Сигналы класса:

- `void not_performed()` – сигнал о сбое копирования;
- `void copy_finished()` – сигнал о завершении копирования.

2.11 Класс ThreadToRemove

Данный класс создан для выполнения удаления системных объектов в отдельном потоке.

Методы класса:

- `void rec_remove(QDir&, SysElem*, SysElem*)` - метод рекурсивного удаления содержимого выбранной папки;
- `void r_move(SysElem*, SysElem*, QString)` – метод выполнения удаления.

Слоты класса:

- `void run_remove(SysElem*, SysElem*, QString, QString)` – слот-обертка для удаления в отдельном потоке.

Сигналы класса:

- `void not_performed()` – сигнал о сбое удаления;
- `void remove_finished()` – сигнал о завершении удаления.

2.12 Класс ThreadToReplace

Данный класс создан для выполнения перемещения системных объектов в отдельном потоке.

Методы класса:

- `void c_py(QDir, SysElem*, SysElem*, QString)` - метод выполнения копирования;
- `void rec_remove(QDir&, SysElem*, SysElem*)` - метод рекурсивного удаления содержимого выбранной папки;
- `void r_move(SysElem*, SysElem*, QString)` – метод выполнения удаления.

Слоты класса:

- `void run_replace(QDir, SysElem*, SysElem*, QString, QString)` – слот-обертка для перемещения в отдельном потоке.

Сигналы класса:

- `void not_performed()` – сигнал о сбое перемещения;
- `void replace_finished()` – сигнал о завершении перемещения.

2.13 Класс ThreadToSearch

Данный класс создан для выполнения поиска системных объектов в отдельном потоке.

Методы класса:

- `void search(QDir&, QString, QFileInfoList&)` - метод выполнения поиска.

Слоты класса:

- `void run_search(QString, QString, QString)` – слот-обертка для поиска в отдельном потоке.

Сигналы класса:

- `void search_finished()` – сигнал о завершении поиска.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1 Алгоритм по шагам слота кнопки создания объекта файловой системы (`void on_btnCreate_clicked()`)

Шаг 1. Получение текущего каталога левой панели файлового менеджера.
Шаг 2. Получение текущего каталога правой панели файлового менеджера.
Шаг 3. Определение панели, на которой была нажата кнопка «Create».
Шаг 4. Проверка на нахождение в каталоге выше каталога пользователя. Если текущий каталог выше каталога пользователя, вызов окна с уведомлением об этом и переход к шагу 17.
Шаг 5. Вызов диалогового окна для выбора типа создаваемого объекта.
Шаг 6. Вызов диалогового окна для ввода имени создаваемого объекта.
Шаг 7. Цикл поиска системного объекта с введенным именем. Если объект с введенным именем найден, вызов окна с уведомлением об этом и переход к шагу 17.
Шаг 8. Получение пути создаваемого объекта.
Шаг 9. Если была выбрана символьная ссылка, переход к шагу 10, иначе переход к шагу 13.
Шаг 10. Вызов диалогового окна для ввода пути связываемого с символьной ссылкой объекта.
Шаг 11. Получение пути связываемого с символьной ссылкой объекта.
Шаг 12. Создание символьной ссылки. Если ссылка не создана, вызов окна с уведомлением об этом и переход к шагу 17.
Шаг 13. Если был выбран файл, переход к шагу 14, иначе переход к шагу 15.
Шаг 14. Создание файла. Если файл не создан, вызов окна с уведомлением об этом и переход к шагу 17.
Шаг 15. Если был выбран каталог, переход к шагу 16, иначе переход к шагу 17.
Шаг 16. Создание каталога. Если каталог не создан, вызов окна с уведомлением об этом.
Шаг 17. Завершение метода.

3.2 Алгоритм по шагам слота кнопки удаления выбранного объекта (`void on_btnRemove_clicked()`)

Шаг 1. Получение текущего каталога левой панели файлового менеджера.
Шаг 2. Получение текущего каталога правой панели файлового менеджера.
Шаг 3. Определение панели, на которой была нажата кнопка «Remove».

Шаг 4. Проверка на нахождение в каталоге выше каталога пользователя. Если текущий каталог выше каталога пользователя, вызов окна с уведомлением об этом и переход к шагу 11.
Шаг 5. Проверка на случай, если не выбран ни один объект. Если не выбран ни один объект, вызов окна с уведомлением об этом и переход к шагу 11.
Шаг 6. Вызов окна с просьбой подтвердить либо отменить удаление. Если было получено подтверждение, переход к шагу 8.
Шаг 7. Если удаление было отменено, производится очистка пути выбранного файла и каталога и переход к шагу 11.
Шаг 8. Блокировка кнопки удаления.
Шаг 9. Отправка сигнала потоку для удаления.
Шаг 10. Очистка пути выбранного файла и каталога.
Шаг 11. Завершение метода.

3.3 Алгоритм по шагам слота кнопки копирования выбранного объекта (`void on_btnCopy_clicked()`)

Шаг 1. Получение текущего каталога левой панели файлового менеджера.
Шаг 2. Получение текущего каталога правой панели файлового менеджера.
Шаг 3. Проверка на нахождение в каталоге выше каталога пользователя. Если текущий каталог выше каталога пользователя, вызов окна с уведомлением об этом, очистка пути выбранного файла и каталога и переход к шагу 12.
Шаг 4. Проверка на случай, если не выбран ни один объект. Если не выбран ни один объект, вызов окна с уведомлением об этом и переход к шагу 12.
Шаг 5. Создание переменной `data` типа `QFileInfo`.
Шаг 6. Если был выбран файл, инициализация переменной `data` через путь выбранного файла и переход к шагу 8.
Шаг 7. Если был выбран каталог, инициализация переменной `data` через путь выбранного каталога.
Шаг 8. Цикл поиска системного объекта с именем, хранящимся в переменной `data`. Если объект с введенным именем найден, вызов окна с уведомлением об этом, очистка пути выбранного файла и каталога и переход к шагу 12.
Шаг 9. Блокировка кнопки копирования.
Шаг 10. Отправка сигнала потоку для копирования.
Шаг 11. Очистка пути выбранного файла и каталога.
Шаг 12. Завершение метода.

3.4 Алгоритм по шагам слота кнопки перемещения выбранного объекта (`void on_btnReplace_clicked()`)

Шаг 1. Получение текущего каталога левой панели файлового менеджера.
Шаг 2. Получение текущего каталога правой панели файлового менеджера.
Шаг 3. Проверка на нахождение в каталоге выше каталога пользователя. Если текущий каталог выше каталога пользователя, вызов окна с уведомлением об этом, очистка пути выбранного файла и каталога и переход к шагу 12.
Шаг 4. Проверка на случай, если не выбран ни один объект. Если не выбран ни один объект, вызов окна с уведомлением об этом и переход к шагу 12.
Шаг 5. Создание переменной `data` типа `QFileInfo`.
Шаг 6. Если был выбран файл, инициализация переменной `data` через путь выбранного файла и переход к шагу 8.
Шаг 7. Если был выбран каталог, инициализация переменной `data` через путь выбранного каталога.
Шаг 8. Цикл поиска системного объекта с именем, хранящимся в переменной `data`. Если объект с введенным именем найден, вызов окна с уведомлением об этом, очистка пути выбранного файла и каталога и переход к шагу 12.
Шаг 9. Блокировка кнопки перемещения.
Шаг 10. Отправка сигнала потоку для перемещения.
Шаг 11. Очистка пути выбранного файла и каталога.
Шаг 12. Завершение метода.

3.5 Алгоритм по шагам метода переименования выбранного объекта (`void on_btnRename_clicked()`)

Шаг 1. Получение текущего каталога левой панели файлового менеджера.
Шаг 2. Получение текущего каталога правой панели файлового менеджера.
Шаг 3. Определение панели, на которой была нажата кнопка «Rename».
Шаг 4. Проверка на нахождение в каталоге выше каталога пользователя. Если текущий каталог выше каталога пользователя, вызов окна с уведомлением об этом и переход к шагу 15.
Шаг 5. Проверка на случай, если не выбран ни один объект. Если не выбран ни один объект, вызов окна с уведомлением об этом и переход к шагу 15.
Шаг 6. Вызов диалогового окна для ввода нового имени.

Шаг 7. Проверка на случай, если имя не введено. Если объект с введенным именем найден, вызов окна с уведомлением об этом и переход к шагу 15.
Шаг 8. Цикл поиска системного объекта с введенным именем.
Шаг 9. Получение нового пути объекта.
Шаг 10. Если был выбран файл, переход к шагу 11, иначе переход к шагу 13.
Шаг 11. Переименование файла. Если файл не был переименован, вызов окна с уведомлением об этом.
Шаг 12. Переход к шагу 15.
Шаг 13. Если был выбран каталог, переход к шагу 14, иначе переход к шагу 15.
Шаг 14. Переименование каталога. Если файл не был переименован, вызов окна с уведомлением об этом.
Шаг 15. Завершение метода.

3.6 Блок-схема метода `void on_lvLeft_clicked()`

Данный метод отвечает за получение пути системного объекта, на который произвели нажатие.

Блок-схема данного метода приведена в приложении Г.

3.7 Блок-схема метода `void on_lvLeft_doubleClicked()`

Данный метод предназначен для открытия системного объекта посредством двойного нажатия.

Блок-схема данного метода приведена в приложении Г.

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Интерфейс программы представлен на рисунке Б.1.

В верхней части главного окна программы представлено поле «Info». При нажатии на него пользователь может ознакомиться с основными комбинациями клавиш для выполнения функций, представленных в программе (см. рисунок Б.2). Также при нажатии на какую-либо запись в поле «Info» указанная в этой записи функция выполнится.

Ниже представлен виджет вкладок. Выполнять создание и удаление вкладок, а также перемещение по вкладкам пользователь может как вручную, так и с помощью комбинаций клавиш, представленных в поле «Info».

Под виджетом вкладок указаны кнопки с иконками, изображающими возможные операции над файлом или каталогом. Хотя иконки интуитивно понятны, при наведении на кнопки с ними всплывают подсказки, поясняющие работу каждой кнопки (см. рисунок Б.3).

Если во время выполнения программа обработает какое-либо исключение, появится предупреждение об этом (см. рисунок Б.4).

Вышеуказанный пример лишь иллюстрация одного из обработанных исключений. Подробно осмотреть возможные исключения можно непосредственно в коде программы.

При нажатии на кнопку для создания появится окно для выбора типа создаваемого объекта: файла, символьной ссылки или каталога (см. рисунок Б.5).

При выборе одного из предложенных типов появится окно для указания имени создаваемого объекта (см. рисунок Б.6).

Чтобы совершить операции удаления и переименования системного объекта, необходимо выбрать его в любой панели отображения файловой системы и нажать на кнопку требуемой функции (либо использовать соответствующую комбинацию клавиш).

Рекомендуется обратить внимание на то, что реализованная программа выполняет удаление в обход Корзины, то есть вернуть удаленные данные невозможно!

При нажатии на кнопку переименования объекта появится окно для ввода нового имени (см. рисунок Б.6).

Рекомендуется обратить внимание на то, что, если нужно ввести имя файла, необходимо указать его расширение (желательно то же, что и было до переименования).

Для выполнения копирования и перемещения системного объекта, необходимо на правой панели отображения файловой системы перейти в каталог, в который объект будет перемещен, после выбрать объект в левой панели и нажать на кнопку требуемой функции (либо использовать соответствующую комбинацию клавиш).

Ниже кнопок основных операций над системными объектами располагаются строки, отражающие текущее местонахождение пользователя в файловой системе в левой и правой панелях отображения файловой систем. В эти строки пользователь имеет возможность вписать путь для перехода в каталог по этому пути.

По центру главного окна программы расположены панели, отображающие файловую систему. С помощью данных панелей осуществляется движение по файловой системе путем двойного клика по ее элементам.

Элементы навигации по каталогам «.» и «..» реализованы программным способом. Они выполняют переходы в корневой и родительский каталоги соответственно.

Ниже панелей отображения файловой системы расположены основные свойства выбранного объекта. Свойства состоят из даты последнего изменения выбранного объекта, его размера и типа.

В нижней части главного окна программы расположена строка для поиска системных объектов по имени. Как и в случае с переименованием, для поиска вместе с названием файла необходимо указать и его расширение. В противном случае будет выполнен поиск по каталогам.

После нажатия на кнопку «Search» она перейдет в состояние блокировки до тех пор, пока не появится окно с результатами поиска (см. рисунок Б.7). Сделано это для того, чтобы пользователь во время выполнения поиска в отдельном потоке не смог нажать на поиск еще раз, так как поток поиска занят, и очередной вызов этого потока ничего не даст.

Также действуют и кнопки копирования, перемещения и удаления (из-за того, что эти операции могут быть длительными, они выполняются в отдельных потоках).

В результате программа позволяет выполнять все основные манипулирования системными объектами: создание, удаление, копирование, перемещение, переименование, поиск, отображение основных свойств, визуальное представление.

ЗАКЛЮЧЕНИЕ

При выполнении курсового проекта были изучены такие темы, как применение объектно-ориентированного программирования при создании приложений, реализация графической оболочки приложения при помощи фреймворка Qt, система сигналов и слотов в среде разработки Qt Creator.

Реализованный двухпанельный файловый менеджер с вкладками хоть и имеет понятный интерфейс и довольно удобный функционал, выполняет основные операции с файлами, все же может быть доработан и улучшен в различных направлениях. Такими направлениями могут быть:

- Возможность множественного выбора системных объектов.
- Оперирование элементами интерфейса методом Drag’n’Drop.
- Возможность отмены уже совершенных действий.
- Синхронизация с облачным хранилищем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Страуструп, Б. Язык программирования C++/ Б.Страуструп; специальное издание. Пер. с англ. — СПб.: BHV, 2008. – 1098 с.

[2] Лафоре, Р. Объектно-ориентированное программирование в C++, 4-е издание / Р. Лафоре — СПб.: Питер, 2004.

[3] Официальный сайт документации Qt [Электронный ресурс]. – Режим доступа - https://doc.qt.io/ – Дата доступа: 11.05.2023

[4] Назначение и виды файловых менеджеров [Электронный ресурс]. – Режим доступа - https://studfile.net/preview/9508732/page:8/ – Дата доступа: 11.05.2023

ПРИЛОЖЕНИЕ А

(*обязательное*)

Листинг программы с комментариями

ПРИЛОЖЕНИЕ Б
(*обязательное*)

Скриншоты работы программы

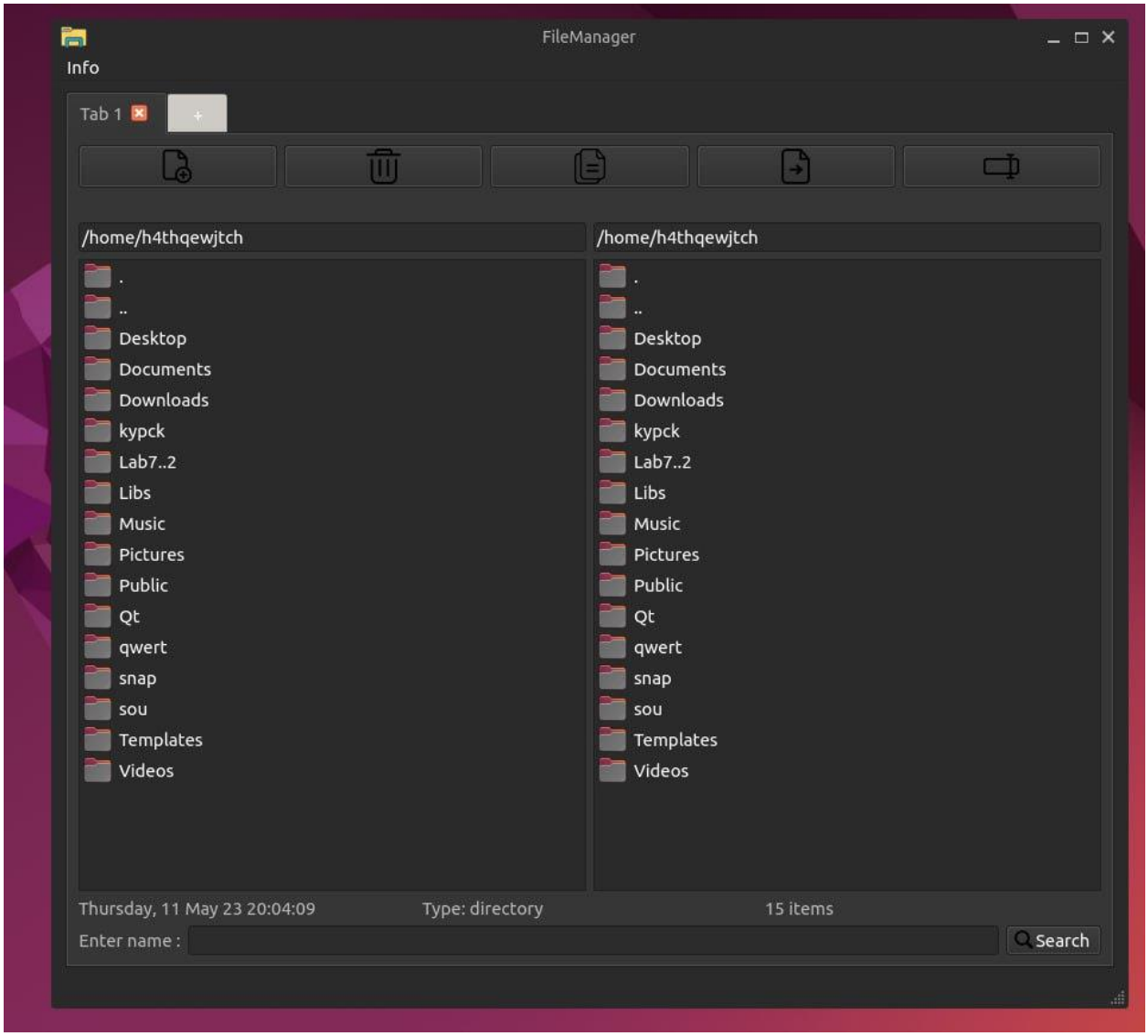

Рисунок Б.1

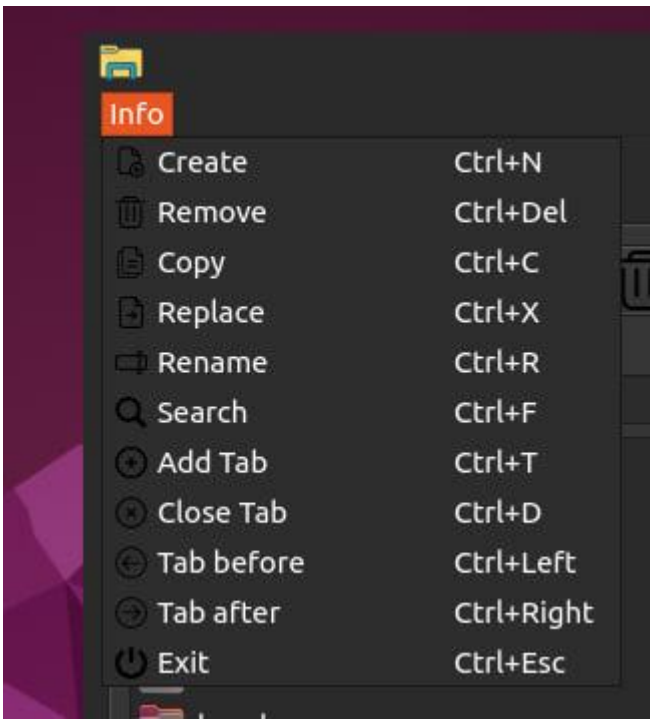

Рисунок Б.2

Рисунок Б.3

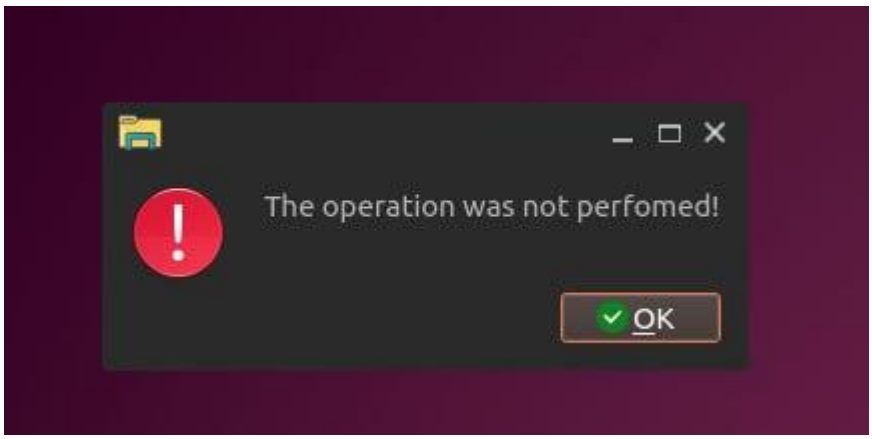

Рисунок Б.4

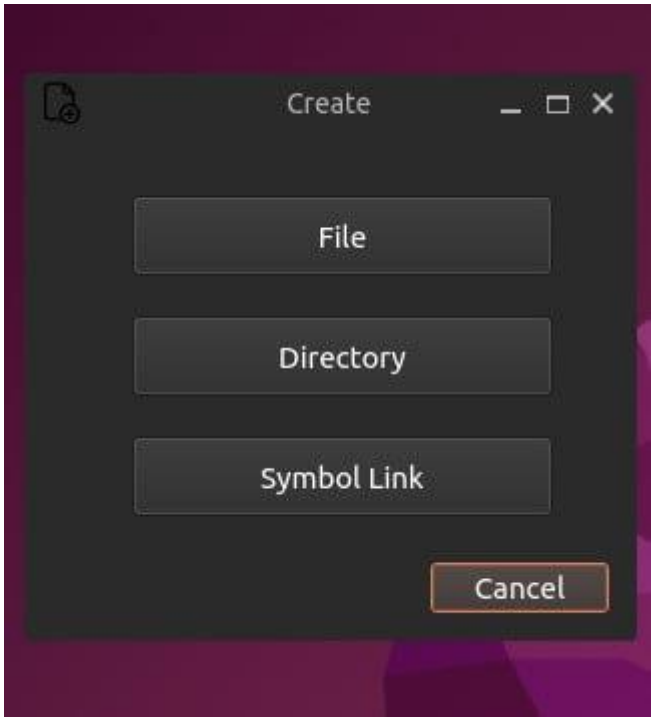

Рисунок Б.5

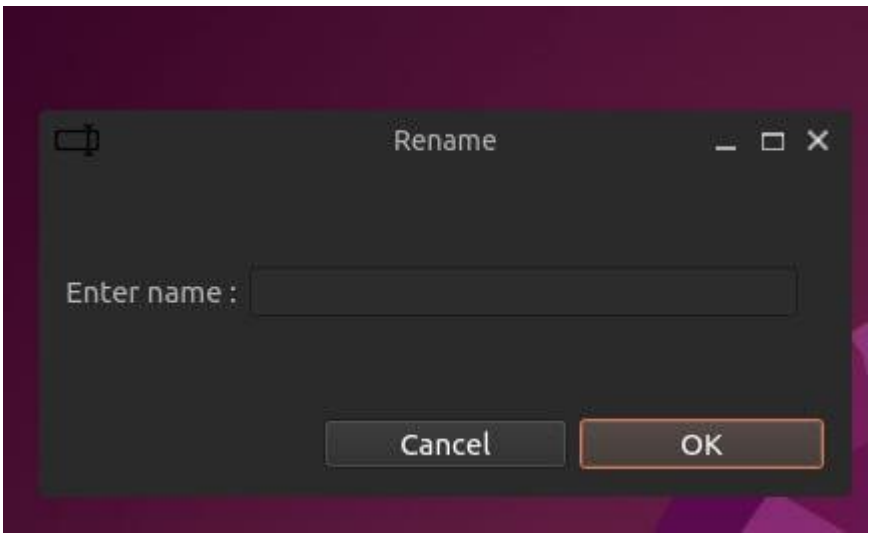

Рисунок Б.6

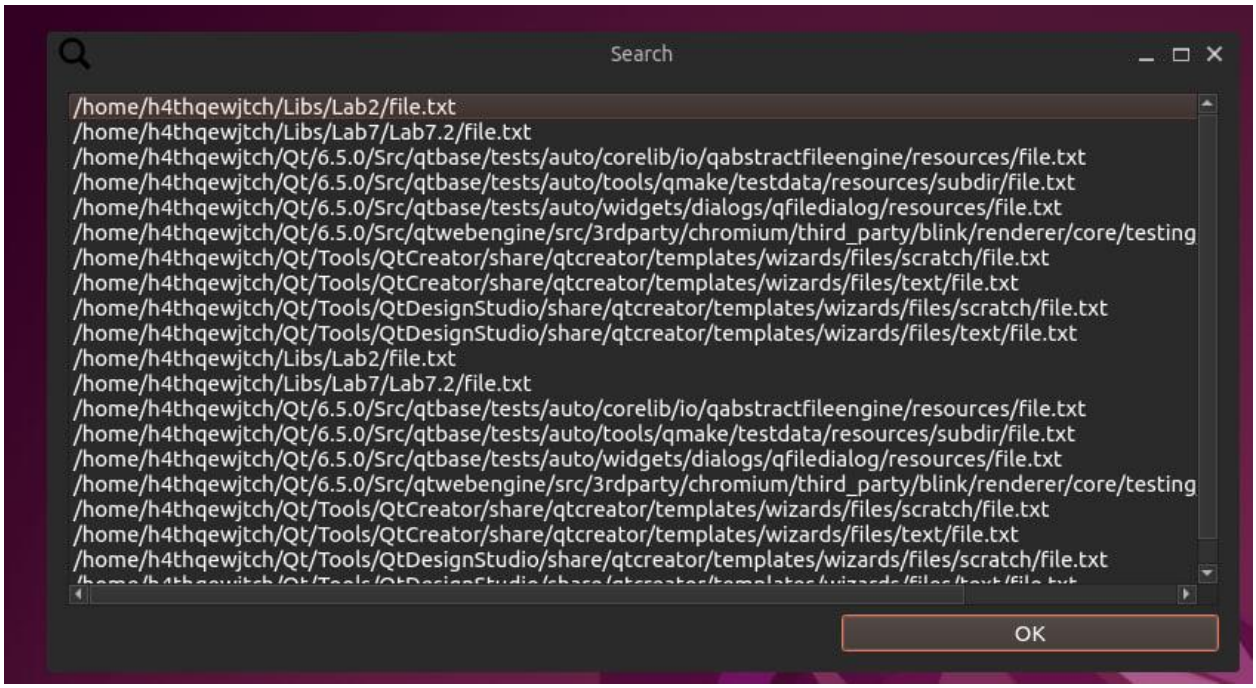

Рисунок Б.7

ПРИЛОЖЕНИЕ В
(о*бязательное*)

Диаграмма классов

ПРИЛОЖЕНИЕ Г
(о*бязательное*)

Блок-схемы алгоритмов

ПРИЛОЖЕНИЕ Д
(о*бязательное*)

Ведомость документов