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

ВВЕДЕНИЕ 4
1 ОБЗОР ЛИТЕРАТУРЫ 5
2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ 6
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 12
4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 14
ЗАКЛЮЧЕНИЕ 16
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17
ПРИЛОЖЕНИЕ А 18
ПРИЛОЖЕНИЕ Б 19
ПРИЛОЖЕНИЕ В 23
ПРИЛОЖЕНИЕ Г 24
ПРИЛОЖЕНИЕ Д 25

ВВЕДЕНИЕ

Объектно-ориентированное программирование (ООП) – методология программирования, основанная на представлении программы в виде совокупности взаимодействующих объектов, являющихся экземплярами определённых классов, каждый из которых представляет собой часть иерархии наследования. Основными принципами объектно-ориентированного программирования являются абстракция (контекстное понимание предмета, формализуемое в виде класса), инкапсуляция (упаковывание данных и поведения в обособленные части единого компонента), наследование (концепция, позволяющая абстрактному типу данных наследовать данные и функционал существующего абстрактного типа данных) и полиморфизм (способность функции обрабатывать данные разных типов).

Среди многих современных языков программирования язык C++ выделяется ориентацией на объектно-ориентированное программирование.

C++ — компилируемый статически типизированный язык программирования общего назначения. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. В сравнении с его предшественником — языком C — наибольшее внимание уделено поддержке объектно-ориентированного и обобщенного 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.

1 ОБЗОР ЛИТЕРАТУРЫ

Файловый менеджер — программа, предоставляющая интерфейс пользователя для работы с файловой системой. Файловый менеджер позволяет выполнять наиболее частые операции над файлами — создание, открытие/проигрывание/просмотр, редактирование, перемещение, переименование, копирование, удаление, отображение атрибутов и свойств, поиск файлов и др. Помимо основных функций, многие файловые менеджеры включают ряд дополнительных возможностей, например, таких, как резервное копирование, управление принтерами и пр.

Выделяют следующие типы файловых менеджеров:

Навигационные (пространственные). 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. Nautilus File Manager является представителем пространственных файловых менеджеров.

Двухпанельные (ортодоксальные). В общем случае имеют две равноценные панели для списка файлов, дерева каталогов и т. п. К ортодоксальным файловым менеджерам относятся: Norton Commander, Total Commander и др.

Консольные. 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.

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++.

Qt 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).

2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

Диаграмма классов UML представлена в ПРИЛОЖЕНИИ В.

Для создания навигационного файлового менеджера в программе были реализованы различные классы в зависимости от требуемого функционала.

2.1 Класс *CreateChoice*

Реализует работу всплывающего окна для выбора системного объекта при его создании.

Атрибуты класса:

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

Методы класса:

- *get_file ()* - метод, сообщающий о выборе создании файла;
- g*et_dir ()* - метод, сообщающий о выборе создания каталога;
- g*et_link ()* - метод, сообщающий о выборе создания каталога.

Слоты класса:

- *on_btnFile_clicked ()* - слот выбора файла;
- *on_btnDir_clicked ()* - слот выбора каталога;
- *on_btnLink_clicked ()* – слот выбора символьной ссылки;
- *on_btnCancel_clicked ()* - слот нажатия на *"Cancel"*.

2.2 Класс *Form*

Отображает внешний вид и работу основного окна приложения.

Атрибуты класса:

- **ui* - указатель для связи с соответствующим ui-файлом;
- **model* - указатель модели данных файловой системы;
- **threadCopy* – указатель на объект для копирования в отдельном потоке;
- **threadRemove* - указатель на объект для удаления в отдельном потоке;
- **threadReplace* - указатель на объект для перемещения в отдельном потоке;
- **threadSearch* - указатель на объект для поиска в отдельном потоке;
- **view* – указатель на объект класса *QListView*;
- **window* – указатель на объект класса *SearchResult*;

- **thCopy* – указатель на объект потока копирования;
- **thRemove* – указатель на объект потока удаления;
- **thReplace* – указатель на объект потока перемещения;
- **thSearch* – указатель на объект потока поиска;
- *f* - объект класса *File* для выполнения операций с текстовыми файлами;
- *d* - объект класса *Dir* для выполнения операций с каталогами;
- **file* - указатель на объект класса *SysElem*;
- **dir* - указатель на объект класса *SysElem*;
- *filePath* – переменная для хранения пути файла;
- *dirPath* - переменная для хранения пути каталога;
- *searchName* – переменная для хранения имени искомого системного объекта.

Методы класса (методы-обертки для слотов):
- *btn_create ();*
- *btn_remove ();*
- *btn_copy ();*
- *btn_replace ();*
- *btn_rename ();*
- *btn_search ().*

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

Сигналы класса:

- *start_copy (QDir, SysElem*, SysElem*, QString, QString)* – сигнал о выполнении копирования*;*
- *start_remove (SysElem*, SysElem*, QString, QString)* – сигнал о выполнении удаления;
- *start_replace (QDir, SysElem*, SysElem*, QString, QString)* – сигнал о выполнении перемещения*;*
- *start_search (QString, QString, QString)* – сигнал о выполнении поиска.

2.3 Класс *LinkedPath*

Создан для записи пути связываемого с символьной ссылкой системного объекта во всплывающем окне.

Атрибуты класса:

- **ui* - указатель для связи с соответствующим ui-файлом;
- *path* - переменная для хранения нового имени выбранного файла.

Методы класса:

- *get_path ()* - метод передачи пути.

Слоты класса:

- *on_btnCancel_clicked ()* – слот нажатия на *"Cancel"*;
- *on_btnOK_clicked()* - слот нажатия на *"OK"*;
- *on_path_textEdited(const QString&)* – слот для ввода пути связываемого файла;

2.4 Класс *MainWindow*

Содержит функционал для взаимодействия с интерфейсом основного окна приложения.

Атрибуты класса:

- **ui* - указатель для связи с соответствующим ui-файлом;
- **form* – указатель на объект класса *Form;*
- **thSearch* – указатель на объект потока поиска;
- **window* – указатель на объект класса *SearchResult*;

Слоты класса:

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

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

2.5 Класс *NewName*

Создан для возможности переименования системного объекта во всплывающем окне.

Атрибуты класса:

- **ui* - указатель для связи с соответствующим ui-файлом;
- *name* - переменная для хранения нового имени выбранного файла.

Методы класса:

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

Слоты класса:

- *on_name_textEdited (const QString&)* - слот ввода нового имени;
- *on_btnOK_clicked ()* - слот нажатия на *“OK”*;
- *on_btnCancel_clicked ()* - слот нажатия на *"Cancel"*.

2.6 Класс *SearchResult*

Реализует поиск системного объекта по имени, указанном в главном окне приложения в соответствующей текстовой строке.

Атрибуты класса:

- **ui* - указатель для связи с соответствующим ui-файлом.

Методы класса:

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

Слоты класса:

- *on_btnOK_clicked()* - слот нажатия на *"OK"*.
- *on_leftList_itemClicked(QListWidgetItem*)* – слот нажатия на панель *leftList*.

2.7 Класс *System*

Данный класс, предназначенный для работы с системными объектами, является абстрактным базовым классом для двух производных от него классов: *File* и *Dir*.

Методы класса:

- *create ()* – чисто виртуальный метод создания системного объекта;
- *r_move ()* – чисто виртуальный метод удаления системного объекта;
- *r_name (QString, QString)* – чисто виртуальный метод переименования системного объекта;
- с*_py (QString)* - чисто виртуальный метод копирования системного объекта.

2.8 Класс *File*

Класс предназначен для работы с текстовыми файлами.

Методы класса:

- *create ()* – метод создания системного объекта;
- *r_move ()* – метод удаления системного объекта;
- *r_name (QString, QString)* – метод переименования системного объекта;
- с*_py (QString)* - метод копирования системного объекта.

2.9 Класс *Dir*

Класс предназначен для работы с каталогами.

Методы класса:

- *create ()* – метод создания системного объекта;
- *r_move ()* – метод удаления системного объекта;
- *r_name (QString, QString)* – метод переименования системного объекта;
- с*_py (QString)* - метод копирования системного объекта.

2.10 Класс *ThreadToCopy*

Создан для выполнения копирования в отдельном потоке.

Методы класса:

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

Слоты класса:

- *run_copy (QDir, SysElem*, SysElem*, QString, QString)* - метод обертка для копирования в отдельном потоке.

Сигналы класса:

- *not_performed ()* – сигнал о сбое копирования;

- *copy_finished ()* – сигнал о завершении копирования.

2.11 Класс *ThreadToRemove*

Создан для выполнения удаления в отдельном потоке.
Методы класса:
- *rec_remove (QDir&, SysElem*, SysElem*)* - метод рекурсивного удаления содержимого выбранной папки;
- *r_move (SysElem*, SysElem*, QString)* – метод выполнения удаления.

Слоты класса:
- *run_remove (SysElem*, SysElem*, QString, QString)* - метод обертка для удаления в отдельном потоке.

Сигналы класса:
- *not_performed ()* – сигнал о сбое удаления;
- *remove_finished ()* – сигнал о завершении удаления.

2.12 Класс *ThreadToReplace*

Создан для выполнения перемещения в отдельном потоке.
Методы класса:
- *c_py (QDir, SysElem*, SysElem*, QString)* - метод выполнения копирования;
- *rec_remove (QDir&, SysElem*, SysElem*)* - метод рекурсивного удаления содержимого выбранной папки;
- *r_move (SysElem*, SysElem*, QString)* – метод выполнения удаления.

Слоты класса:
- *run_replace (QDir, SysElem*, SysElem*, QString, QString)* - метод обертка для перемещения в отдельном потоке.

Сигналы класса:
- *not_performed ()* – сигнал о сбое перемещения;
- *replace_finished ()* – сигнал о завершении перемещения.

2.13 Класс *ThreadToSearch*

Создан для выполнения перемещения в отдельном потоке.
Методы класса:
- *search (QDir&, QString, QFileInfoList&)* - метод выполнения поиска.

Слоты класса:
- *run_search (QString, QString, QString)* - метод обертка для поиска в отдельном потоке.

Сигналы класса:
- *search_finished ()* – сигнал о завершении поиска.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1 Алгоритм по шагам метода создания объекта файловой системы (o*n_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 выбрана функция *“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 19.
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 19.
Шаг 8. Получение пути создаваемого объекта.
Шаг 9. Если была выбрана символьная ссылка, переход к шагу 10, иначе переход к шагу 14.
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.
Шаг 13. Переход к шагу 19.
Шаг 14. Если был выбран файл, переход к шагу 15, иначе переход к шагу 17.
Шаг 15. Создание файла. Если файл не создан, вызов окна с уведомлением об этом.
Шаг 16. Переход к шагу 19.
Шаг 17. Если был выбран каталог.
Шаг 18. Создание каталога. Если каталог не создан, вызов окна с уведомлением об этом.
Шаг 19. Завершение метода.

3.2 Алгоритм по шагам метода переименования выбранного объекта (*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 выбрана функция *“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.

Шаг 6. Вызов диалогового окна для ввода нового имени.

Шаг 7. Проверка на случай, если имя не введено.

Шаг 8. Цикл поиска системного объекта с введенным именем. Если объект с введенным именем найден, вызов окна с уведомлением об этом и переход к шагу 15.

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.

3.3 Блок-схема метода *on_lvSource_clicked()*

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

Блок-схема данного метода приведена в приложении Г.

3.4 Блок-схема метода *on_lvSource_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 *“.”* 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, 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, все же показан как пример и не является пределом совершенства, 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: 11.12.2022

ПРИЛОЖЕНИЕ А

(*обязательное*)

Листинг программы с комментариями

ПРИЛОЖЕНИЕ Б
(*обязательное*)

Скриншоты работы программы

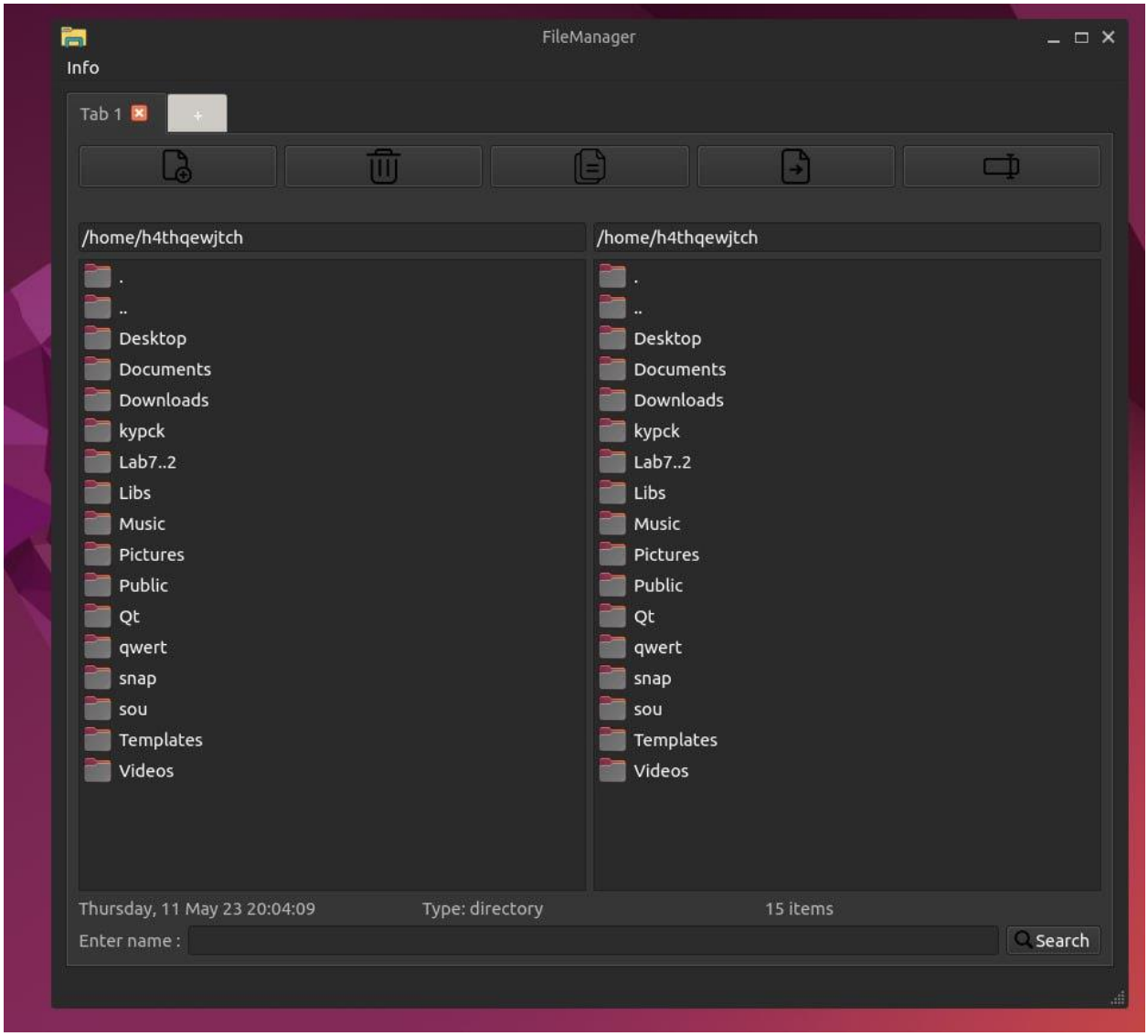

Рисунок Б.1

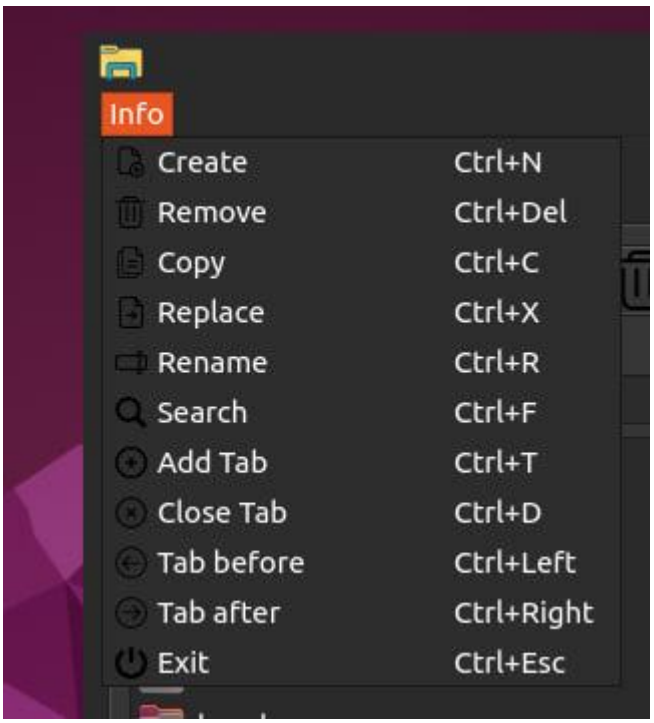

Рисунок Б.2

Рисунок Б.3

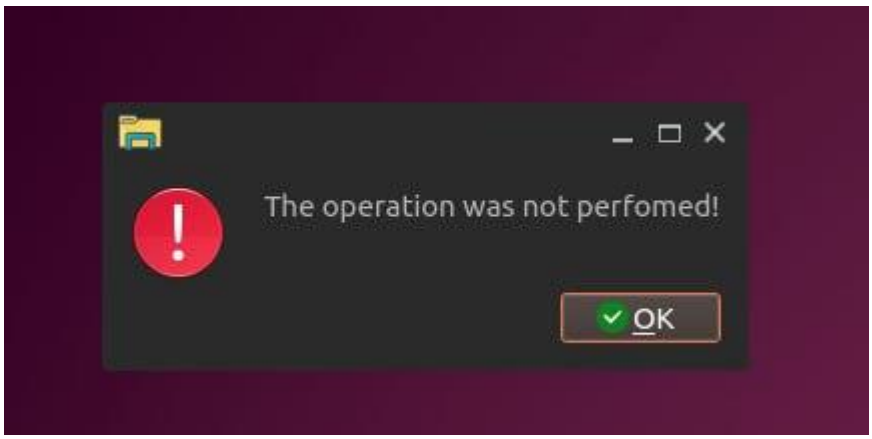

Рисунок Б.4

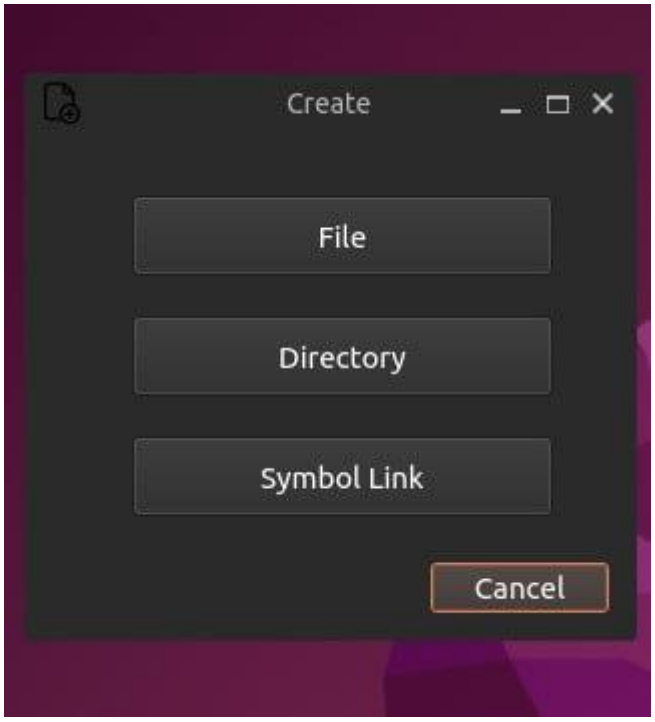

Рисунок Б.5

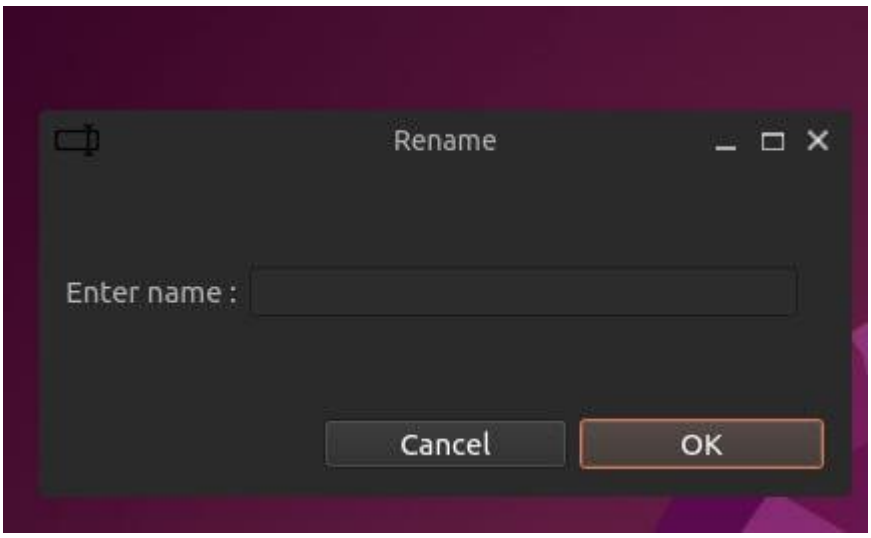

Рисунок Б.6

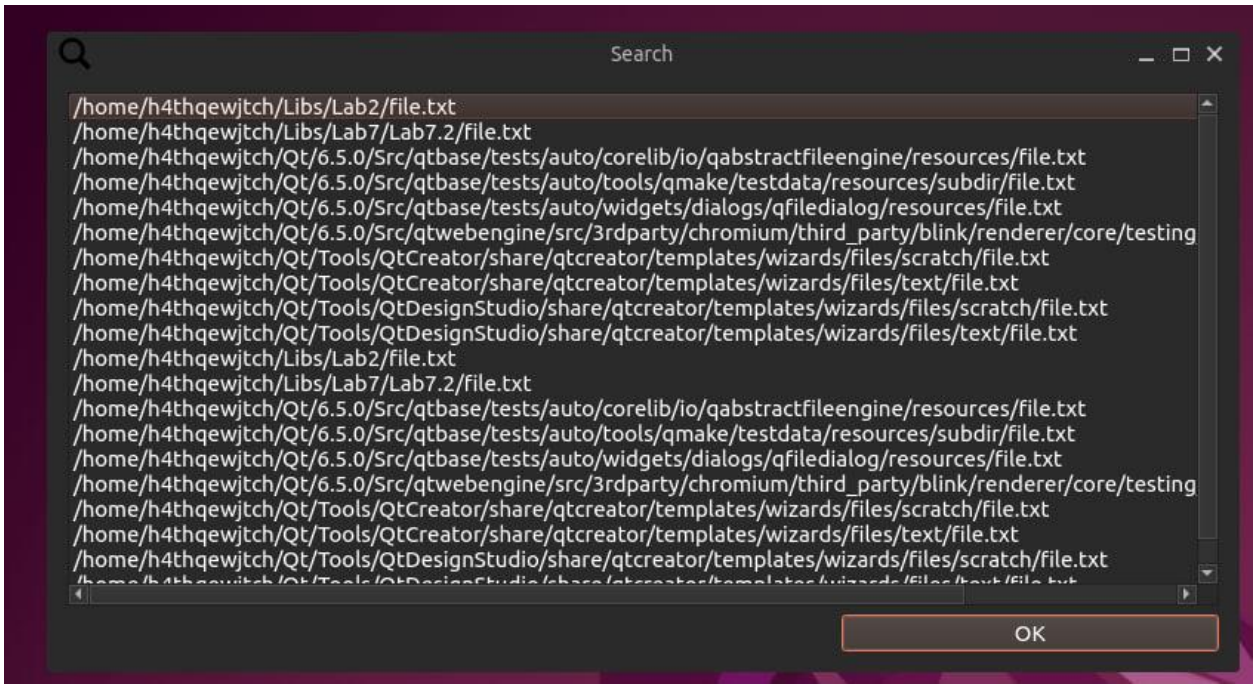

Рисунок Б.7

ПРИЛОЖЕНИЕ В
(о*бязательное*)

Диаграмма классов

ПРИЛОЖЕНИЕ Г
(о*бязательное*)

Блок-схемы алгоритмов

ПРИЛОЖЕНИЕ Д
(о*бязательное*)

Ведомость документов