Кафедра «Высшая математика»

Е.В. Кадура

Операционные   
системы

Методические указания   
по выполнению курсовой работы

Хабаровск

2020

Оглавление

[ВВЕДЕНИЕ 3](#_Toc57384807)

[1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 4](#_Toc57384808)

[1.1 Обзор основных инструментов разработки 4](#_Toc57384809)

[1.2 Концепция «Модель-представление» в Qt. 5](#_Toc57384810)

[2 РЕАЛИЗАЦИЯ ПРОГРАММЫ-ДИСПЕТЧЕРА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++. 6](#_Toc57384811)

[2.1 Назначение разработки 6](#_Toc57384812)

[2.2 Описание интерфейса создаваемого приложения 6](#_Toc57384813)

[2.3 Описание системы меню 7](#_Toc57384814)

[2.4 Описание вспомогательных форм 8](#_Toc57384815)

[3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 9](#_Toc57384816)

[3.1 Общие сведения о программном продукте 9](#_Toc57384817)

[3.2 Описание установки 9](#_Toc57384818)

[3.3 Описание запуска 9](#_Toc57384819)

[3.4 Инструкции по работе 9](#_Toc57384820)

[ЗАКЛЮЧЕНИЕ 11](#_Toc57384821)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 12](#_Toc57384822)

[ПРИЛОЖЕНИЕ 1. ИСХОДНЫЙ КОД ПРОГРАММЫ 14](#_Toc57384823)

# ВВЕДЕНИЕ

Целью курсовой работы является закрепление изученного материала в ходе разработки программы – диспетчера операционной системы Windows, позволяющей просматривать текстовые файлы, расположенные в дереве каталогов файловой системы локальных дисков.

Разработка приложения будет осуществляться в среде Microsoft Visual Studio на языке программирования C++, с использованием платформы программирования Qt для построения графического пользовательского интерфейса. Выбор языка и платформы обусловлен необходимостью освоить совмещение популярной среды среды разработки Microsoft Visual Studio и мощной сторонней библиотеки классов Qt, значительно превосходящей по возможностям и удобству использования библиотеку MFC.

# ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

## Обзор основных инструментов разработки

Для создания приложения в этой работе использованы следующие инструменты:

1. компилятор Microsoft C++.
2. среда разработки Microsoft Visual Studio 2017 (Community edition).
3. платформа разработки кроссплатформенных приложений Qt.

Самый быстрый способ создания приложений под Windows – это, безусловно, современный язык программирования C#, органично интегрированный в среду разработки приложений Visual Studio.

Однако более сложные вызовы, стоящие перед разработчиком, зачастую бывают и более интересными, поэтому в этой работе было принято решение временно отложить C# в сторону и обратить взор на другой язык программирования, C++ компиляция на котором также доступна при работе в Visual Studio.

Существует довольно ограниченный набор инструментов быстрой разработки приложений с графическим интерфейсом на C++ для ОС Windows. Сюда можно причислить не самую популярную библиотеку MFC (Microsoft Foundation Classes), изначально интегрированную в Visual Studio, а также активно развивающуюся платформу Qt.

Этот инструмент разработки изначально был создан в виде библиотеки классов для проектирования пользовательского интерфейса, но уже давно перерос своё первичное предназначение и превратился в мощную кроссплатформенную (предназначенную для компиляции исходного кода для выполнения под управлением различных операционных систем) библиотеку разработки приложений.

Вместе с Qt можно обычно устанавливают редактор кода QtCreator, который на самом деле, так же как и сама библиотека, давно эволюционировал в мощную среду разработки, подобную Visual Studio.

Дискуссия о сравнении этих двух инструментов разработки выходит далеко за рамки данной работы, но задача о совмещении платформы Qt и среды разработки Visual Studio – эта возможность появилось в Visual Studio 2010 – выглядит достаточно интересной, что и послужило причиной выбора инструментов для разработки приложения в этой работе.

Платформа программирования Qt, наряду с множеством других инструментов, содержит сокращённую реализацию парадигмы MVC (Model – View – Controller). Эта концепция в том или ином в виде реализована в разных программных платформах и используется для создания трёхуровневых приложений, в которых модель данных (Model) и её визуальное представление (View) отделены друг от друга и взаимодействуют между собой под управлением третьего компонента (Controller).

Для создания графического интерфейса пользователя библиотекой Qt предоставляется архитектура Model – View, в которой для простоты использования устранён промежуточный агент, так что управление взаимодействием данных и интерфейса лежит на исходном коде самого приложения.

## Концепция «Модель-представление» в Qt.

Библиотека Qt представляет множество классов для гибкой настройки отображения самых разных данных в различных элементах визуального отображения – списки, таблицы, графики и т.д. – например, [QAbstractItemModel](http://qt-project.org/doc/qt-5.0/qtcore/qabstractitemmodel.html), [QAbstractListModel](http://qt-project.org/doc/qt-5.0/qtcore/qabstractlistmodel.html), [QAbstractTableModel](http://qt-project.org/doc/qt-5.0/qtcore/qabstracttablemodel.html) и другие. Однако Qt предоставляет несколько готовых простых моделей для некоторых случаев: [QStringListModel](http://qt-project.org/doc/qt-5.0/qtcore/qstringlistmodel.html), [QDirModel](http://qt-project.org/doc/qt-5.0/qtwidgets/qdirmodel.html), [QFileSystemModel](http://qt-project.org/doc/qt-5.0/qtwidgets/qfilesystemmodel.html), [QSqlQueryModel](http://qt-project.org/doc/qt-5.0/qtsql/qsqlquerymodel.html), которые уже содержат большую часть готового функционала и избавляют программиста от написания значительной части повторяющегося кода [11].

В программе, реализуемой в данной работе, для синхронизации дерева папок и файлов с элементами отображения будут использоваться экземпляры класса QFileSystemModel.

# РЕАЛИЗАЦИЯ ПРОГРАММЫ-ДИСПЕТЧЕРА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++.

## Назначение разработки

Разрабатываемое приложение позволит осуществлять навигацию в рамках локальной файловой системы и просматривать текстовое содержимое содержащихся в её каталогах файлов.

Планируется добавить возможность сохранения настроек между сеансами работы.

Главное меню программы включает два корневых элемента - «Файл» и «Справка».

Меню «Файл»:

* действие «Сохранить»;
* флаг «Автосохрание при выходе»;
* флаг «Отображать все файлы».

«Справка»:

* действие «О программе».

Для связи элементов меню и панели инструментов используем объекты класса QAction, представляющие собой обработчик действий пользователя.

Для связи структуры файловой системы с элементами графического отображения и управления (QTreeView и QListView) используем экземпляры класса QFileSystemModel.

Для сохранения и восстановления настроек приложения между сеансами работы используем экземпляр класса QSettings, позволяющий управлять настройками, хранимыми либо в реестре Windows, либо в ini-файле. В нашем случае настройки будут храниться в разделе реестра «HK\_CURRENT\_USER\Software\Dispatcher\TextViewer».

## Описание интерфейса создаваемого приложения

Главное окно приложения содержит следующие элементы управления (Рис. 2.1):

* строка меню;
* панель инструментов;
* поле редактирования фильтров
* окно отображения дерева каталогов
* окно отображения списка файлов из выбранного каталога;
* окно отображения содержимого выбранного файла в текстовом виде.

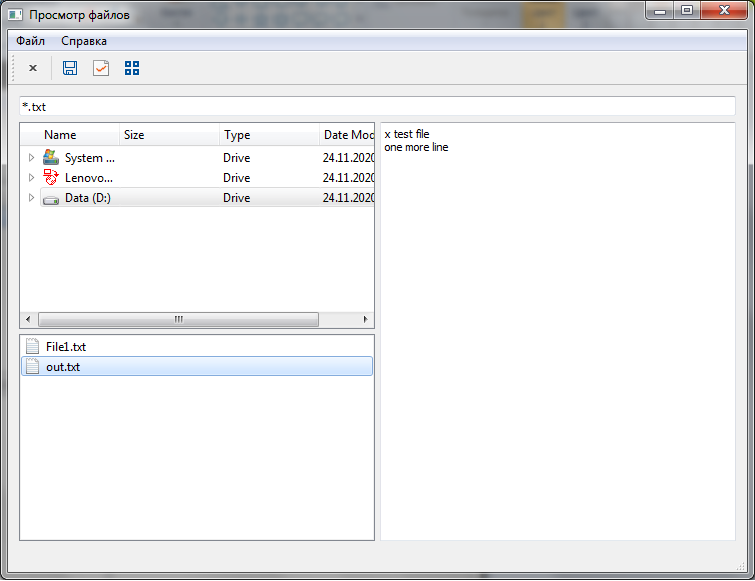


Рисунок 2.1. Главное окно приложения.

В верхней части окна расположена строка меню, состоящая из двух корневых элементов, «Файл» и «Справка».

Все команды меню «Файл» дублируются в виде иконок на панели инструментов, расположенной чуть ниже.

Между панелью инструментов и окнами представления данных находится тестовое поле, в котором указываются фильтры по имени файлов для сокращения множества отображаемых данных – например, можно установить фильтр по расширению «\*.txt». Фильтров может быть несколько, они должны отделяться символом «;» (точка с запятой).

## Описание системы меню

Главное меню программы включает два корневых элемента следующего содержания:

1. «Файл»:
   1. действие «Сохранить» – сохраняет текущие параметры сеанса работы;
   2. флаг «Автосохрание при выходе» – указывает: нужно ли сохранить параметры текущего сеанса работы при завершении работы;
   3. флаг «Отображать все файлы» – при включенном состоянии, файлы, имена которых не подпадают ни под один из указанных фильтров, всё равно будут отображены блеклым серым цветом (в противном случае они показаны не будут вообще).
2. «Справка»:
   1. действие «О программе» – выводит на экран модальный диалог с краткой информацией о приложении.

## Описание вспомогательных форм

Пункт меню «Справка» – «О программе» выводит на экран модальное диалоговое окно с названием и годом создания программы (Рис. 2. 2).

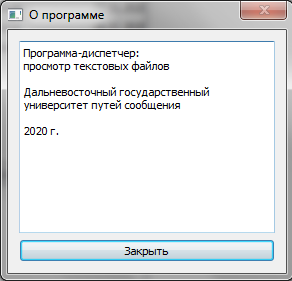


Рисунок 2.2. Вид диалогового окна «О программе».

Для закрытия окна и возврата в главное окно приложения необходимо нажать соответствующую кнопку в нижней части диалога.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Общие сведения о программном продукте

Приложение «Просмотр файлов» позволяет отображать текстовое представление содержимого файлов, хранящихся в локальной файловой системе. Отображение древовидной структуры файловой системы сделано в несколько изменённом стиле «Проводника» Windows – дерево каталогов отображается в верхнем левом окне, список файлов выбранного каталога – в левом нижнем окне (Рис. 2.1).

Содержимое конкретного файла отображается в отдельном окне в правой части основной формы приложения, при щелчке по его имени в списке файлов. Программа заранее не знает природу данных, хранящихся в выбранном файле, но в любом случае выводит любые хранящиеся в нём данные в виде текста в наиболее популярной кодировке “UTF-8”.

## Описание установки

Установка приложения не требуется, однако для его работы должна быть предустановленна среда разработки Microsoft Visual Studio.

## Описание запуска

Приложение запускается из среды Microsoft Visual Studio. Для запуска необходимо открыть проектное решение texted.sln выбрать пункт меню «Отладка – Запуск без отладки» или «Отладка – Начать отладку».

При наличии изменений в оригинальном коде, возможны ошибки компиляции, в этом случае необходимо обратиться к разработчику.

## Инструкции по работе

Сразу после запуска приложения на экране отобразится главное окно, позволяющее воспользоваться всеми реализованными в программе функциями.

В верхней части главного окна приложения расположена строка меню с пунктами «Файл» и «Справка», которые в виде дочерних элементов содержат все доступные пользователю действия.

Меню «Файл» содержит следующие элементы:

* действие «Сохранить»;
* флаг «Автосохрание при выходе»;
* флаг «Отображать все файлы».

«Справка»:

* действие «О программе».

Ниже расположена панель инструментов, в которой продублированы действия, содержащиеся в меню «Файл».

Под панелью инструментов находится текстовое поле, содержащее текущие фильтры по имени файла. Фильтры отделяются друг от друга сиволом «;» (точка с запятой). Редактирование этого поля будет влиять на список файлов текущего каталога.

Ещё ниже, в левой части окна представлен список имеющихся в системе корневых разделов и по умолчанию после запуска выбран первый элемент.

Навигация по каталогам реализована по образу и подобию стандартного проводника Windows, поэтому при выборе нужного каталога в дереве представления, в левом нижнем окне отображается список файлов, находящихся в этом каталоге. Этот список ограничен фильтрами по имени файла, указанными в соответствующем поле. При этом ограничение может быть двух видов – отбракованные файлы либо не отображаются совсем, либо отображаются светло-серым (как бы «затенённым») цветом – в этом случае пользователь в любом случае, независимо от фильтров, видит всё содержимое выбранного каталога, при этом интересующие его файлы отображаются чёрным цветом, а остальные – светло-серым. Для управления указанным ограничением предназначен пункт меню «Файл» – «Все файлы», а дублирующий его элемент панели инструментов.

Справа от «Проводника» находится элемент отображения содержимого выбранного файла. Содержимое выводится в текстовом виде, то есть любые данные интерпретируются как текст в кодировке «UTF-8».

По умолчанию, после завершения работы все настройки теряются, то есть если пользователь изменял фильтры или изменял режим отображения («Все файлы»), то при следующем запуске приложения все настройки будут восстановлены к предыдущему состоянию.

Сохранить текущие настройки можно вручную, выбрав пункт меню «Файл» – «Сохранить». В этом случае при перезапуске программы настройки будут теми же, что перед её закрытием. Кроме того, если установить флажок «Файл» – «Автосохранение при выходе», все параметры будут сохраняться при закрытии приложения автоматически. К сохраняемым параметрам относится и сам флажок «Автосохранение при выходе».

# ЗАКЛЮЧЕНИЕ

В результате работы была спроектировано и разработано приложение – диспетчер ОС windows с функциями навигации по локальной файловой системе и просмотра текстового содержимого файлов.

В ходе работы над проектом получены практические навыки разработки программ на языке высокого уровня C++. Изучено множество аспектов разработки программ на этом языке – в первую очередь, использование среды разработки Visual Studio и платформы программирования Qt, а также особенности использования парадигмы «Модель – Представление» при разработке пользовательской интерфейса. Кроме того, изучена работа с сохранением и загрузкой настроек приложения с помощью встроенных механизмов Qt.

На основе практического применения закреплены базовые теоретические знания, полученные ранее.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Павловская, Т. А. C/C++. Программирование на языке высокого уровня : учеб. для вузов / Т.А. Павловская. – СПб. : Питер, 2011. – 464 с.

2. Саммерфилд, М. Qt. Профессиональное программирование.Разработка кроссплатформенных приложений на C++ / М. Саммерфилд. - М.: Символ-Плюс, 2011. - 560 c.

3. Роберт, С. Сикорд Безопасное программирование на C и C++ / Роберт С. Сикорд. - Москва: РГГУ, 2014. - 496 c.

4. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум, Т. Остин. – 6-е изд. – СПб. : Питер, 2015. – 816 с.

5. Понамарев, В. Программирование на C++/C# в Visual Studio .NET 2003 / В. Понамарев. - М.: БХВ-Петербург, 2015. – 917 c.

6. Кениг, Э. Эффективное программирование на C++. Практическое программирование на примерах. Т. 2 / Э. Кениг, Б.Э. Му. - М.: Вильямс, 2016. - 368 c.

7. Хенкеманс, Д. Программирование на C++ / Д. Хенкеманс, М. Ли. - СПб.: Символ-плюс, 2015. - 416 c.

8. Шлее, М. Qt 5.3. Профессиональное программирование на C++. В подлиннике / М. Шлее. - СПб.: BHV, 2016. – 928 c.

9. [www.stackoverflow.com](http://www.stackoverflow.com) – Популярный интернет-ресурс для обмена опытом разработчиками ПО.

10. [www.ravesli.com/uroki-po-qt5](http://www.ravesli.com/uroki-po-qt5) – русскоязычный учебник по Qt.

11. [www.habr.com/ru/post/171443](http://www.habr.com/ru/post/171443) – статья «Использование моделей в Qt».

12. [www.itnotesblog.ru/note.php?id=159](http://www.itnotesblog.ru/note.php?id=159) – статья «Модель-представления в Qt».

13. [www.qtdoc.ru](http://www.qtdoc.ru) – русское Qt-сообщество.

14. [www.qtcentre.org](http://www.qtcentre.org) – англоязычный форум, посвящённый программированию с использованием Qt.

15. [doc.qt.io](http://www.doc.qt.io) – официальная документация Qt.

16. ГОСТ 2.105-95 Общие требования к текстовым документам.

**ПРИЛОЖЕНИЯ**

# ПРИЛОЖЕНИЕ 1. ИСХОДНЫЙ КОД ПРОГРАММЫ

aboutdialog.h

#pragma once

#include <QDialog>

class AboutDialog : public QDialog

{

public:

AboutDialog(QWidget\* owner = nullptr);

};

aboutdialog.cpp

#include "aboutdialog.h"

#include <QTextEdit>

#include <QPushButton>

#include <QGridLayout>

AboutDialog::AboutDialog(QWidget\* owner)

: QDialog(owner, Qt::WindowTitleHint | Qt::WindowSystemMenuHint)

{

setWindowTitle("О программе");

QTextEdit\* txt = new QTextEdit();

txt->setReadOnly(true);

txt->append("Программа-диспетчер:");

txt->append("просмотр текстовых файлов");

txt->append("");

txt->append("Дальневосточный государственный университет путей сообщения");

txt->append("");

txt->append("2020 г.");

QPushButton\* cmd = new QPushButton("Закрыть");

connect(cmd, SIGNAL(clicked()), SLOT(accept()));

QGridLayout\* layout = new QGridLayout(this);

layout->addWidget(txt, 0, 0);

layout->addWidget(cmd, 1, 0);

setLayout(layout);

}

texted.h

#pragma once

#include "ui\_texted.h"

#include <QtWidgets/QMainWindow>

#include <QFileSystemModel>

#include <QFile>

#include <QSettings>

class texted : public QMainWindow

{

Q\_OBJECT

public:

texted(QSettings& settings, QWidget \*parent = Q\_NULLPTR);

~texted();

private slots:

// обработчики событий пользовательского интерфейса

void on\_dirsView\_clicked(const QModelIndex &index);

void on\_filesView\_clicked(const QModelIndex &index);

void on\_actSave\_triggered();

void on\_actAbout\_triggered();

void on\_actShowAll\_toggled(bool arg1);

void on\_actAutosave\_toggled(bool arg1);

void on\_txtFilter\_editingFinished();

private:

// загрузить настройки сессии

void loadSession();

// сохранить настройки сессии

void saveSession();

// синхронизация представлений каталогов и файлов

void syncViews(const QString& path);

// объект для загрузки пользовательского графического

// интерфейса из xml-файла mainwindow.ui

Ui::textedClass ui;

// модели данных для навигации по файловой системе

QFileSystemModel\* dirsModel;

QFileSystemModel\* filesModel;

// объект файла для чтения содержимого текстовых файлов

QFile file;

// параметры сессии

// Структура для хранения имени, значения по умолчанию

// и текущего значения параметра

template<class T> struct StoredParam

{

QString name;

const T defaultValue;

T currentValue;

StoredParam(const QString &name, const T defVal)

: name(name), defaultValue(defVal), currentValue(defVal)

{

}

};

// ссылка на объект для сохранения/восстановления

QSettings& settings;

// Фильтр по имени файлов

StoredParam<QStringList> filter{ "filter", {"\*.txt", "\*.log"} };

// Показовать только отфильтрованные файлы

StoredParam<bool> showAllFiles{ "showAllFiles", false };

// Автосохранение сессии при выходе

StoredParam<bool> autosave{ "autosave", false };

};

texted.cpp

#include "texted.h"

#include "aboutdialog.h"

#include <QTextStream>

#include <QTextCodec>

#include <QToolBar>

#include <QLineEdit>

texted::texted(QSettings& settings, QWidget \*parent)

: QMainWindow(parent), settings(settings)

{

// загрузка пользовательского интерфейса

ui.setupUi(this);

// восстановление сохранённых настроек

loadSession();

// подключаем объекты действий к слотам

connect(ui.actQuit, &QAction::triggered, qApp, QApplication::quit);

// помещаем действия в меню "Файл"

ui.menuFile->addAction(ui.actSave);

ui.menuFile->addAction(ui.actAutosave);

ui.menuFile->addAction(ui.actShowAll);

ui.menuFile->addSeparator();

ui.menuFile->addAction(ui.actQuit);

// помещаем действия в меню "Справка"

ui.menuHelp->addAction(ui.actAbout);

// помещаем действия на панель инструментов

QToolBar\* tbMain = addToolBar("main toolbar");

tbMain->addAction(ui.actQuit);

tbMain->addSeparator();

tbMain->addAction(ui.actSave);

tbMain->addAction(ui.actAutosave);

tbMain->addAction(ui.actShowAll);

QString root = "C:/";

// создаём модель данных для каталогов и присоединяем к ней окно каталогов

dirsModel = new QFileSystemModel(this);

dirsModel->setFilter(QDir::NoDotAndDotDot | QDir::AllDirs);

dirsModel->setRootPath(root);

ui.dirsView->setModel(dirsModel);

// создаём модель данных для файлов и присоединяем к ней окно файлов

filesModel = new QFileSystemModel(this);

filesModel->setFilter(QDir::NoDotAndDotDot | QDir::Files);

filesModel->setNameFilters(filter.currentValue);

filesModel->setNameFilterDisables(showAllFiles.currentValue);

ui.filesView->setModel(filesModel);

syncViews(root);

// визуализация параметров сессии

ui.txtFilter->setText(filter.currentValue.join(";"));

ui.actShowAll->setChecked(showAllFiles.currentValue);

ui.actAutosave->setChecked(autosave.currentValue);

}

texted::~texted()

{

if (autosave.currentValue)

{

saveSession();

}

}

void texted::loadSession()

{

// загружаем текущие параметры сессии

filter.currentValue = settings.value(filter.name, filter.defaultValue).toStringList();

showAllFiles.currentValue = settings.value(showAllFiles.name, showAllFiles.defaultValue).toBool();

autosave.currentValue = settings.value(autosave.name, autosave.defaultValue).toBool();

}

void texted::saveSession()

{

// сохраняем текущие параметры сессии

settings.setValue(filter.name, filter.currentValue);

settings.setValue(showAllFiles.name, showAllFiles.currentValue);

settings.setValue(autosave.name, autosave.currentValue);

}

void texted::syncViews(const QString& path)

{

ui.filesView->setRootIndex(filesModel->setRootPath(path));

}

void texted::on\_dirsView\_clicked(const QModelIndex &index)

{

// при щелчке ЛКМ по узлу дерева папок, отображаем его содержимое в окне файлов

syncViews(dirsModel->fileInfo(index).absoluteFilePath());

}

void texted::on\_filesView\_clicked(const QModelIndex &index)

{

// при щелчке по имени файла, читаем его содержимое как текст в кодировке UTF-8 и отображаем

QString fileName = filesModel->fileInfo(index).absoluteFilePath();

file.setFileName(fileName);

if (file.open(QFile::ReadOnly | QFile::Text)) {

ui.textView->clear();

QTextStream text(&file);

text.setCodec(QTextCodec::codecForName("UTF-8"));

while (!text.atEnd()) {

ui.textView->append(text.readLine());

}

file.close();

}

}

void texted::on\_actSave\_triggered()

{

saveSession();

}

void texted::on\_actAbout\_triggered()

{

// Выводим диалог "О программе"

AboutDialog\* dlg = new AboutDialog;

dlg->exec();

delete dlg;

}

void texted::on\_actShowAll\_toggled(bool arg1)

{

showAllFiles.currentValue = arg1;

filesModel->setNameFilterDisables(showAllFiles.currentValue);

}

void texted::on\_actAutosave\_toggled(bool arg1)

{

autosave.currentValue = arg1;

}

void texted::on\_txtFilter\_editingFinished()

{

filter.currentValue = ui.txtFilter->text().split(";");

filesModel->setNameFilters(filter.currentValue);

}

texted.ui

*<?xml* version="1.0" encoding="UTF-8"*?>*

*<ui* version="4.0"*>*

*<class>*textedClass*</class>*

*<widget* class="QMainWindow" name="textedClass"*>*

*<property* name="geometry"*>*

*<rect>*

*<x>*0*</x>*

*<y>*0*</y>*

*<width>*739*</width>*

*<height>*542*</height>*

*</rect>*

*</property>*

*<property* name="windowTitle"*>*

*<string>*texted*</string>*

*</property>*

*<widget* class="QWidget" name="centralWidget"*>*

*<layout* class="QVBoxLayout" name="verticalLayout"*>*

*<item>*

*<widget* class="QLineEdit" name="txtFilter"*>*

*<property* name="toolTip"*>*

*<string>*Фильтры по имени файлов (разделены символом &quot;;&quot;)*</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget* class="QSplitter" name="splitter\_2"*>*

*<property* name="orientation"*>*

*<enum>*Qt::Horizontal*</enum>*

*</property>*

*<widget* class="QSplitter" name="splitter"*>*

*<property* name="orientation"*>*

*<enum>*Qt::Vertical*</enum>*

*</property>*

*<widget* class="QTreeView" name="dirsView"*/>*

*<widget* class="QListView" name="filesView"*/>*

*</widget>*

*<widget* class="QTextEdit" name="textView"*/>*

*</widget>*

*</item>*

*</layout>*

*</widget>*

*<widget* class="QMenuBar" name="menuBar"*>*

*<property* name="geometry"*>*

*<rect>*

*<x>*0*</x>*

*<y>*0*</y>*

*<width>*739*</width>*

*<height>*20*</height>*

*</rect>*

*</property>*

*<widget* class="QMenu" name="menuFile"*>*

*<property* name="title"*>*

*<string>*Файл*</string>*

*</property>*

*</widget>*

*<widget* class="QMenu" name="menuHelp"*>*

*<property* name="title"*>*

*<string>*Справка*</string>*

*</property>*

*</widget>*

*<addaction* name="menuFile"*/>*

*<addaction* name="menuHelp"*/>*

*</widget>*

*<widget* class="QStatusBar" name="statusBar"*/>*

*<action* name="actSave"*>*

*<property* name="icon"*>*

*<iconset>*

*<normaloff>*../../img/save.png*</normaloff>*../../img/save.png*</iconset>*

*</property>*

*<property* name="text"*>*

*<string>*Сохранить*</string>*

*</property>*

*<property* name="toolTip"*>*

*<string>*Сохранить состояние текущей сессии*</string>*

*</property>*

*<property* name="shortcut"*>*

*<string>*Ctrl+S*</string>*

*</property>*

*</action>*

*<action* name="actQuit"*>*

*<property* name="icon"*>*

*<iconset>*

*<normaloff>*../../img/close.png*</normaloff>*../../img/close.png*</iconset>*

*</property>*

*<property* name="text"*>*

*<string>*Выход*</string>*

*</property>*

*<property* name="toolTip"*>*

*<string>*Завершение работы*</string>*

*</property>*

*<property* name="shortcut"*>*

*<string>*Ctrl+Q*</string>*

*</property>*

*</action>*

*<action* name="actAbout"*>*

*<property* name="icon"*>*

*<iconset>*

*<normaloff>*../../img/info.png*</normaloff>*../../img/info.png*</iconset>*

*</property>*

*<property* name="text"*>*

*<string>*О программе*</string>*

*</property>*

*<property* name="toolTip"*>*

*<string>*Информация о приложении*</string>*

*</property>*

*</action>*

*<action* name="actShowAll"*>*

*<property* name="checkable"*>*

*<bool>*true*</bool>*

*</property>*

*<property* name="icon"*>*

*<iconset>*

*<normaloff>*../../img/all.png*</normaloff>*../../img/all.png*</iconset>*

*</property>*

*<property* name="text"*>*

*<string>*Все файлы*</string>*

*</property>*

*<property* name="toolTip"*>*

*<string>*Отображать все отфильтрованные файлы затенёнными*</string>*

*</property>*

*</action>*

*<action* name="actAutosave"*>*

*<property* name="checkable"*>*

*<bool>*true*</bool>*

*</property>*

*<property* name="icon"*>*

*<iconset>*

*<normaloff>*../../img/autosave.png*</normaloff>*../../img/autosave.png*</iconset>*

*</property>*

*<property* name="text"*>*

*<string>*Автосохранение при выходе*</string>*

*</property>*

*</action>*

*</widget>*

*<layoutdefault* spacing="6" margin="11"*/>*

*<resources>*

*<include* location="texted.qrc"*/>*

*</resources>*

*<connections/>*

*</ui>*

main.cpp

#include "texted.h"

#include <QtWidgets/QApplication>

#include <QSettings>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

QSettings settings("Dispatcher", "TextViewer");

texted w(settings);

w.show();

return a.exec();

}