Национальное образовательное частное учреждение дополнительного профессионального образования «Национальный открытый университет «ИНТУИТ»

Допускаю к защите

Первый проректор **А.П. Сериков**

« » 2021 г.

ВЫПУСКНАЯ РАБОТА

на тему **«Разработка программы для апробации библейских переводов»**

Программа профессиональной переподготовки
«Программирование»

Исполнитель: Мигунов А. Ю.

Руководитель: Сериков А. П.

Москва

2021

Введение

Темой данной выпускной работы является разработка программы для апробации библейских переводов.

Апробация — это проверка понимания текста перевода носителями языка, которые не участвовали в работе над переводом, чтобы убедиться, что перевод будет понятен будущим читателям книги. Апробация является одним из этапов работы над переводом Библии, в особенности, при переводе на те языки, на которые Библия переводится впервые.

Цель компьютерной программы, разрабатываемой в данной работе – это создание программы для апробаторов (людей, проводящих апробацию) с удобным интерфейсом, которая позволяет упростить и ускорить составление отчета по апробации и предоставляет ряд других опций, необходимых для работы апробатора.

В 2019 году автором данной работы была создана консольная программа для создания отчетов для апробации на языке C#. Эта программа была создана в текстовом редакторе gedit в Linux и скомпилирована также в Linux, но при этом один и тот же exe файл запускался и в Linux, и в Windows. Таким образом была обеспечена полная кроссплатформенная переносимость.

В данной работе использованы некоторые идеи из той программы, но создаваемая в ней программа является GUI программой с намного более расширенными возможностями.

Глава 1. Постановка задачи и определение требований

Разрабатываемая программа должна, прежде всего, обеспечивать возможность быстрого и простого составления отчета по апробации. Она должна также обеспечивать возможность одновременно открывать все документы, необходимые для проведения апробации: 1) список вопросов для апробации; 2) отчет по апробации; 3) текст перевода, по которому проводится апробация; 4) текст на русском языке для возможности сравнения. Кроме того, программа должна иметь удобный интерфейс и необходимые опции.

Поскольку в процессе работы над переводом Библии используется программа Paratext, разрабатываемая программа должна предоставлять возможность открывать и просматривать файлы, созданные в Paratext, то есть тексты переводимых книг, а также русские переводы. Эти файлы имеют особый формат – USFM (расширение - .SFM), который содержит ряд специальных маркеров для разметки. Для отображения форматирования требуется создать специальный конвертер, пригодный для практических целей. При этом апробаторы не должны изменять файлы Paratext, поэтому они должны открываться только для чтения.

Для создания списков вопросов иногда используется Transcelerator, который является плагином для Paratext. Он генерирует файлы формата HTML. Соответственно, разрабатываемая программа должна обеспечивать возможность открывать такие файлы для чтения.

Разрабатываемая программа должна быть кроссплатформенной – для Windows и для Linux.

Кроме того, желательно обеспечить возможность выбора языка интерфейса – по крайней мере, выбора между русским и английским языком.

Глава 2. Выбор IDE и фреймворка для разработки

Одним из требований для разрабатываемой программы является кроссплатформенная переносимость. Хотя программа разрабатывается на Linux, должна быть обеспечена возможность простого перенесения ее на Windows.

Кроссплатформенных фреймворков и IDE,обеспечивающих простую разработку графического интерфейса, доступных для Linux, существует не очень много. Выбор делался между тремя наиболее распространенными IDE – MonoDevelop, Lazarus и Qt Creator.

MonoDevelop использует графическую библиотеку GTK# и язык C#. Как утверждают разработчики MonoDevelop, он позволяет создавать и компилировать программы, которые могут запускаться на Linux и Windows (без повторной компиляции) и для запуска которых пользователям обеих OS требуется только скачать и установить GTK# на свой компьютер. Поэтому этот вариант первоначально выглядел как наиболее оптимальный. Однако оказалось, что программы, созданные и скомпилированные в MonoDevelop на Linux, не запускаются в Windows после установки GTK#. Кроме того, существует очень мало информации по MonoDevelop даже на английском языке, а на русском она практически отсутствует. Библиотеке GTK# также свойственен недостаток, присущий GTK+ (для языка C), а именно сложность внесения изменений в графический интерфейс. По этим причинам от использования MonoDevelop пришлось отказаться.

Lazarus и Qt Creator, в принципе, позволяют использовать один и тот же программный код на Linux и Windows, но требуют отдельной компиляции программы в соответствующей OS. При более внимательном анализе выяснилось, что у Lazarus всё же возникают проблемы при компиляции на Windows программного кода, созданного в

Linux, в то время, как у Qt проблем практически нет (если не исользовать в программе специфических особенностей Linux, например, таких, как особенности файловой системы). Кроме того, Qt по сравнению с Lazarus предоставляет намного более широкие возможности. Поэтому выбор был сделан в пользу Qt.

Помимо основной версии Qt для C++, существуют также версии для других языков, в частности, PyQt для Python. Этот вариант также является кроссплатформенным. PyQt с Python обладает тем преимуществом, что Python позволяет быстрее писать программный код, чем C++ и требует меньшего объема кода. Однако программы на Python работают намного медленнее, чем на C++. Кроме того, Qt с C++ обеспечивает намного более удобную среду разработки, чем PyQt с Python. При разработке на PyQt с Python графический интерфейс создается в Qt Designer, а для остальной части процесса разработки требуется другая IDE, например, PyCharm. А при разработке на Qt с C++ весь процесс разработки можно вести в Qt Creator, который позволяет создавать и графический интерфейс, и весь программный код. Таким образом, этот вариант является более удобным при разработке, а разрабатываемая программа работает быстрее, поэтому этот вариант был выбран для данной работы.

Qt также предоставляет очень удобные возможности для создания интерфейса на разных языках (с помощью программы Qt Linguist). Это позволяет обеспечить возможность выбора английского или русского интерфейса.

Глава 3. Разработка графического интерфейса программы

3.1. Центральная часть (центральный виджет) главного окна

Апробатору во время проведения апробации нужно иметь возможность работать с четырьмя документами: 1) текст на языке перевода; 2) текст на русском языке (для сравнения); 3) список вопросов для апробации; 4) отчет по апробации. Из них нужно вносить изменения только в отчет по апробации. Остальные документы изменять не нужно.

Таким образом, программа должна открывать четыре документа, три из них только для чтения, а один – для чтения и записи.

Для реализации открытия четырех документов можно использовать MDI (Multiple Document Interface), который поддерживается в Qt. Однако, во-первых, в этом случае будет сложнее контролировать, какие документы в каком режиме должны открываться, а во-вторых, открывать более четырех документов апробатору всё равно не потребуется.

Проще использовать SDI (Single Document Interface), в который встроены четыре многострочных текстовых поля для просмотра каждого из четырех документов. В Qt для этого можно использовать элементы textEdit (текстовое поле) совместно с элементами label (метка) с текстом заголовка для каждого поля.

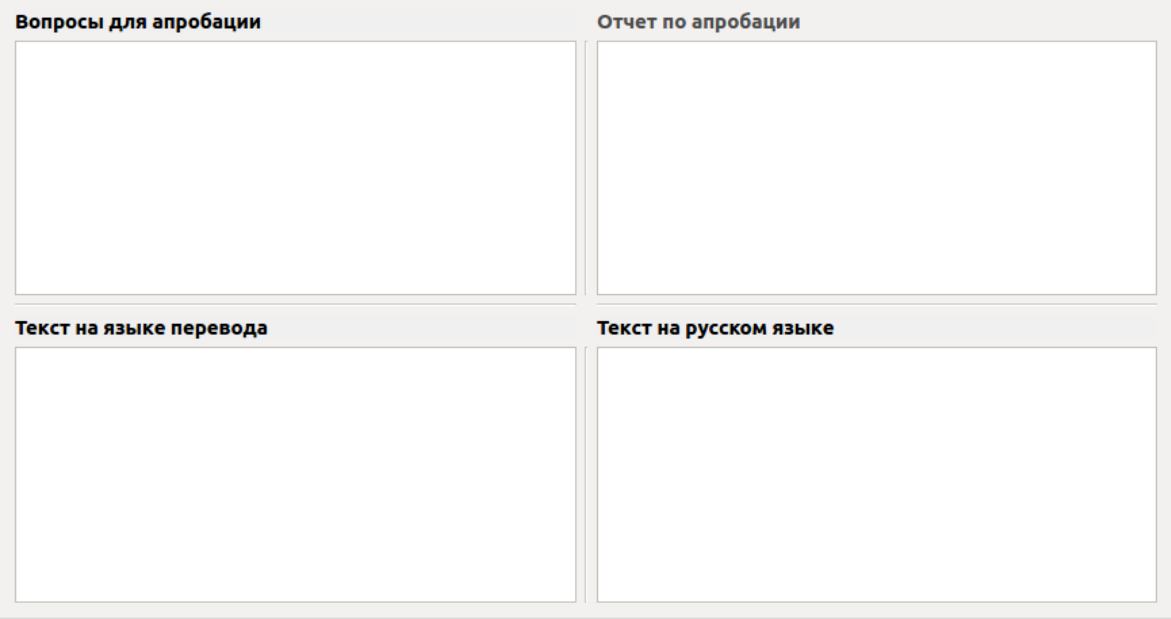

Рис. 1. Центральный виджет главного окна, русский вариант (в Linux)

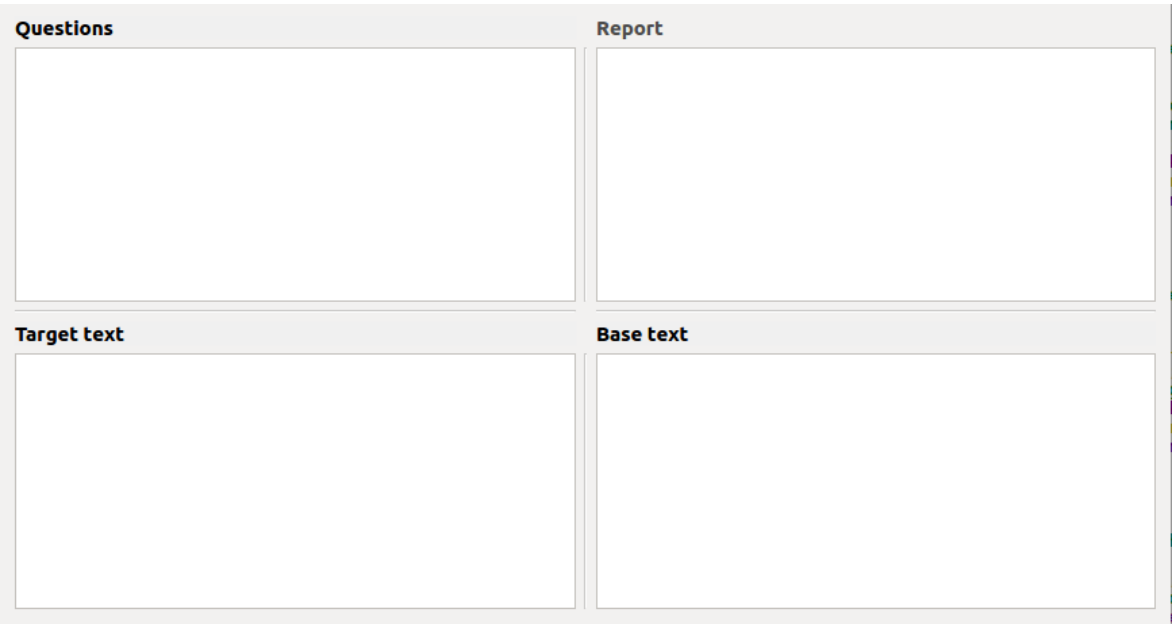

Рис. 2. Центральный виджет главного окна, английский вариант (в Linux)

3.2. Главное меню

В главном меню должна быть предусмотрена возможность выбора всех или, по крайней мере, большинства опций, предоставляемых программой. Некоторые из этих опций будут стандартными, а некоторые – специфичными именно для разрабатываемой программы.

Основные подменю главного меню программы:

- Файл (стандартное меню);
- Редактировать (стандартное меню);
- Вид (стандартное меню);
- Вставка (из этого можно будет автоматически вставить в отчет некоторую служебную информацию, такую как заголовок отчета, время начала и окончания работы и так далее);
- Ответ респондента (здесь будут перечислены возможные ответы респондента, то есть, человека, с которым апробатор проводит апробацию; эти типовые ответы респондента апробатор может автоматически вставить в отчет без необходимости набирать их вручную);
- Комментарий (здесь будут перечислены некоторые возможные комментарии респондента к своему ответу, которые апробатор также может автоматически вставить в отчет);
- Опции (здесь будут некоторые опции для настройки программы);
- Справка (сведения о программе).

Все варианты ответов респондента и комментариев были использованы в консольной программе для апробации и были взяты из нее.

Учитывая, что в программе будет четыре текстовых поля, для некоторых опций главного меню нужно будет предусмотреть выбор одного из четырех полей. С другой стороны, опции, позволяющие

редактировать текст, должны быть доступны только для текстового поля для отчета.

Таким образом, получается следующий список подменю:

1. Файл

1.1. Открыть (4 варианта – для каждого из текстовых полей)

1.2. Закрыть (4 варианта)

1.3. Создать новый файл отчета

1.4. Сохранить отчет

1.5. Сохранить отчет как

1.6. Печать (4 варианта)

1.7. Предварительный просмотр (4 варианта)

1.8. Конвертировать в PDF (4 варианта)

1.9. Конвертировать в ODT (4 варианта)

1.10. Выйти из программы

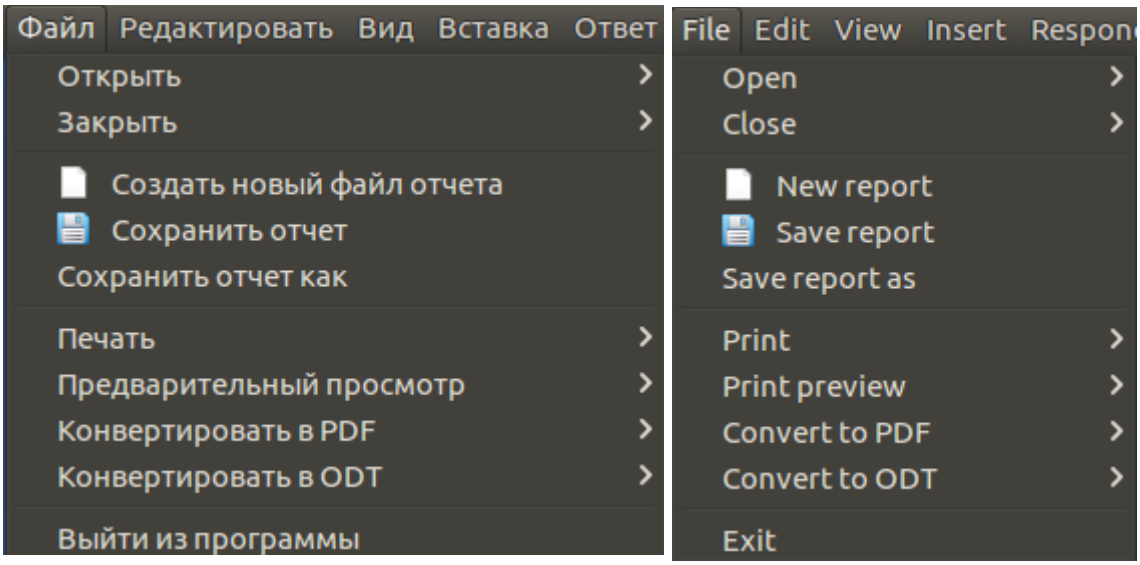

Рис. 3. Подменю “Файл”, русский и английский варианты в Linux

2. Редактировать

2.1. Отменить

2.2. Вернуть

2.3. Вырезать (из отчета)

2.4. Копировать (4 варианта – чтобы обеспечить копирование из каждого из документов)

2.5. Вставить (в отчет)

2.6. Копировать и вставить (4 варианта – эта опция объединяет копирование и вставку и позволяет совершать такую операцию быстрее)

2.7. Удалить (из отчета)

2.8. Выделить всё (4 варианта)

2.9. Найти (4 варианта)

2.10. Найти и заменить (в отчете)

2.11. Найти и заменить везде (в отчете – эта опция позволяет изменять все вхождения слова для поиска на другое за одно действие)

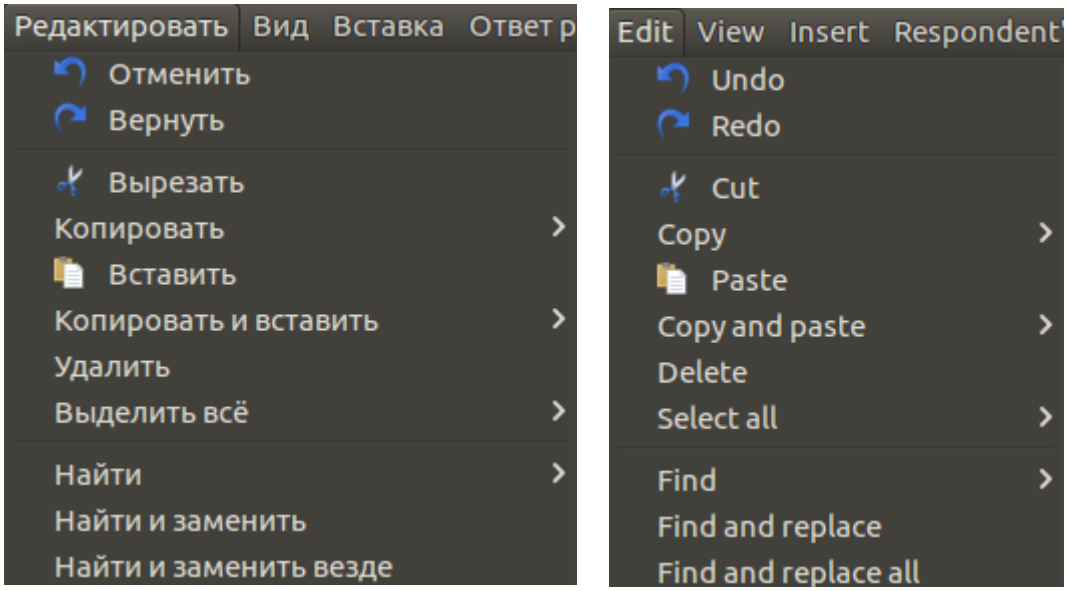

Рис. 4. Подменю “Редактировать”, русский и английский варианты в Linux

3. Вид

3.1. Увеличить (5 вариантов – позволяет увеличить текст во всех текстовых полях или только в одном из них)

3.2. Уменьшить (5 вариантов – позволяет уменьшить текст также во всех текстовых полях или в одном из них)

3.3. Шрифт (4 варианта – позволяет выбрать шрифт в одном из текстовых полей)

3.4. Цвет шрифта (4 варианта)

3.5. Цвет фона текста (4 варианта)

3.6. Цвет фона окна (5 вариантов – позволяет изменить цвет фона в одном из текстовых полей или сразу во всех)

3.7. Текст на языке перевода

3.7.1. Форматированный (USFM / Paratext) (при выборе этой опции текст на языке перевода из файла формата USFM, который используется программой для создания библейских переводов Paratext, будет отформатирован для просмотра в соответствии с тегами USFM)

3.7.2. Неформатированный (USFM / Paratext) (при выборе этой опции текст не будет отформатирован, при этом теги USFM будут отображаться вместе с основным текстом документа)

3.8. Текст на русском языке

3.8.1. Форматированный (USFM / Paratext)

3.8.2. Неформатированный (USFM / Paratext)

3.9. Файл с вопросами (для апробации)

3.9.1. Форматированный (USFM / Paratext)

3.9.2. Неформатированный (USFM / Paratext)

3.10. Кодировка (4 варианта – позволяет изменить кодировку текста в одном из текстовых полей)

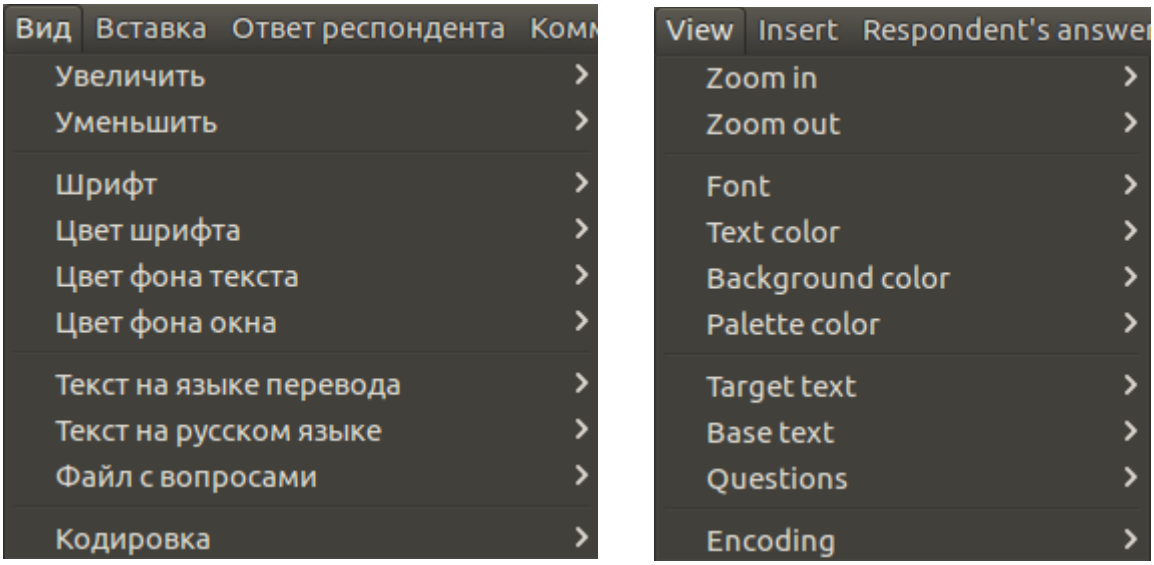

Рис. 5. Подменю “Вид”, русский и английский варианты в Linux

4. Вставка (в отчет)

4.1. Вставить название книги (той книги Библии, по которой проводится апробация)

4.2. Вставить номер

4.2.1. Вставить номер главы (из соответствующей книги Библии)

4.2.2. Вставить номера глав (указывается диапазон глав)

4.2.3. Вставить номер стиха (из соответствующей главы)

4.2.4. Вставить номера стихов (указывается диапазон стихов)

4.3. Вставить номер вопроса (из списка вопросов)

4.4. Вставить время

4.4.1. Вставить время начала работы

4.4.2. Вставить время окончания работы

4.5. Вставить заголовок в отчет (вставляется заголовок отчета)

5. Ответ респондента

5.1. Респондент дал правильный ответ

5.2. Респондент дал правильный ответ, но есть замечания

5.3. Респондент дал частично правильный ответ

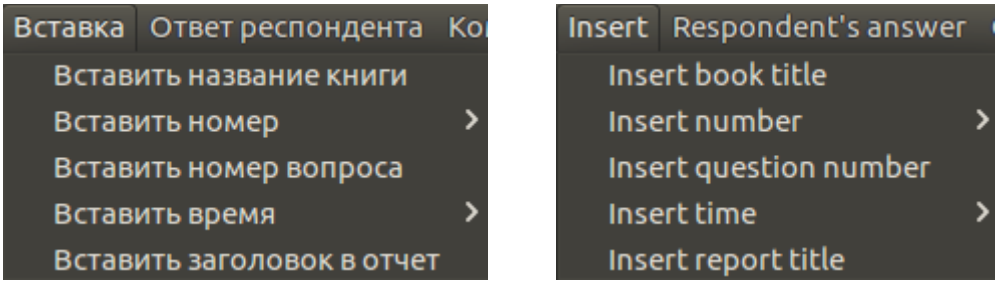

Рис. 6. Подменю “Вставка”, русский и английский варианты в Linux

5.4. Респондент дал частично правильный ответ, и есть замечания

5.5. Респондент дал неправильный ответ

5.6. Респондент дал неправильный ответ, и есть замечания

5.7. Респондент не смог дать ответ

5.8. Респондент не смог дать ответ, и есть замечания

5.9. Апробатор не задавал вопрос, поскольку не было необходимости (такая ситуация возникает, если по предыдущим ответам респондента понятно, что он знает ответ на данный вопрос)

6. Комментарий

6.1. Неизвестное слово, лучше заменить на другое

6.2. Слишком длинное и сложное предложение

6.3. Непонятно, к кому или чему относится местоимение

6.4. Лучше изменить порядок слов в предложении

6.5. Непонятно образное выражение

6.6. Непонятна связь между предложениями

6.7. Непонятна связь между словами (все слова понятны)

6.8. Непонятна основная мысль или вообще о чем идет речь

6.9. В тексте есть что-то неприемлемое для местной культуры

6.10. Респонденту что-то не понравилось в тексте

6.11. Респонденту что-то понравилось в тексте

6.12. Другое замечание или проблема

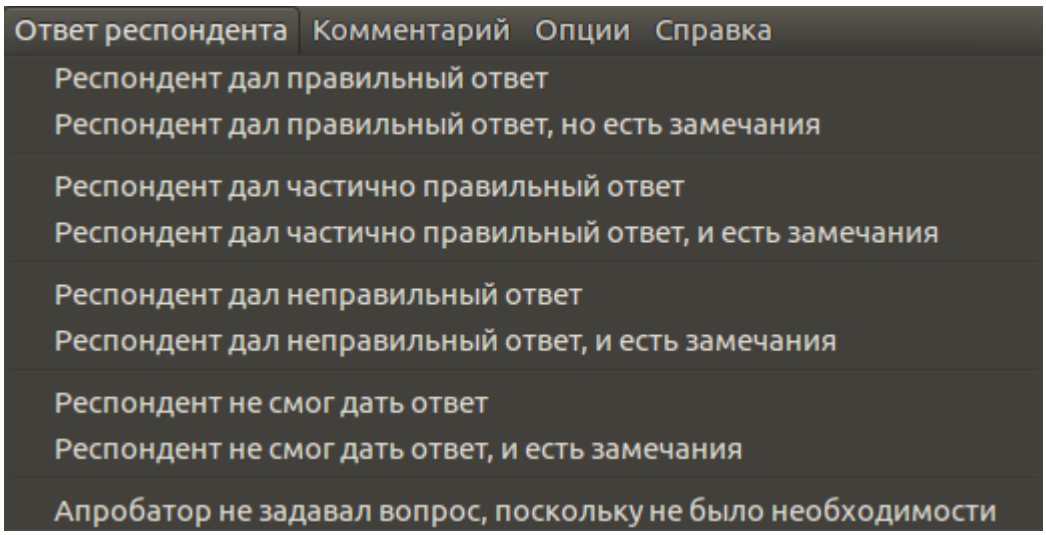

Рис. 7. Подменю “Ответ респондента”, русский вариант в Linux

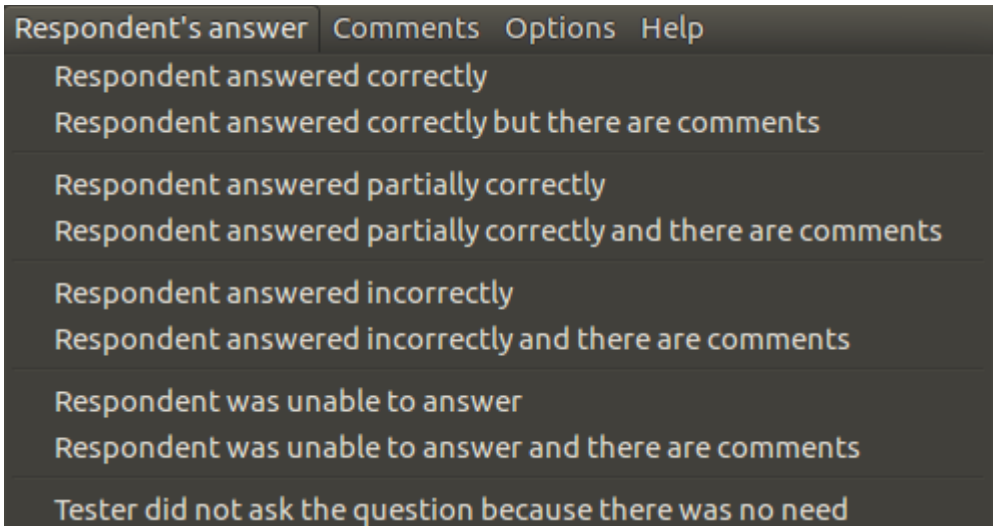

Рис. 8. Подменю “Ответ респондента”, английский вариант в Linux

7. Опции

7.1. Язык интерфейса

7.1.1. Английский

7.1.2. Русский

7.2. Автоматическая вставка времени (эта опция позволяет автоматически вставлять в отчет время начала работы при открытии отчета и время окончания работы при его закрытии)

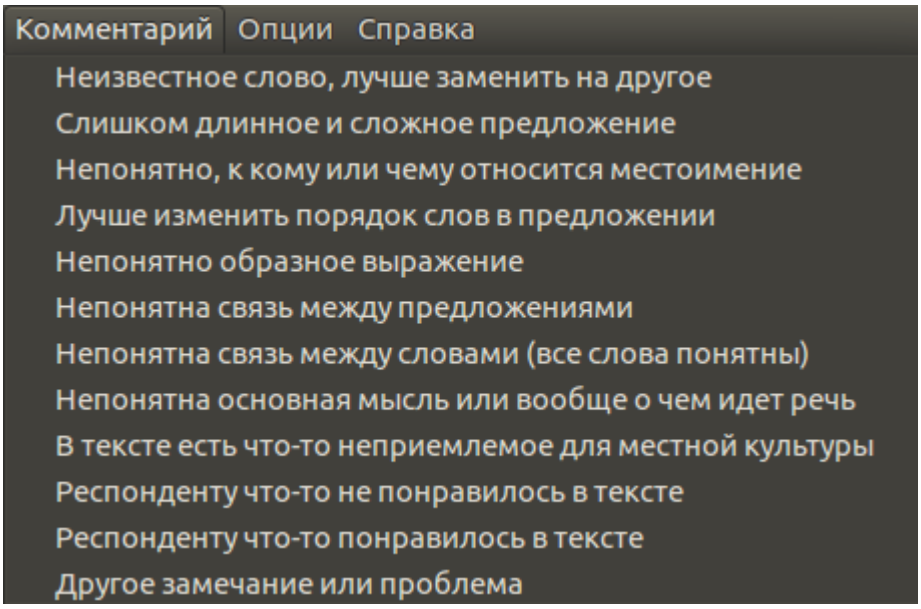

Рис. 9. Подменю “Комментарий”, русский вариант в Linux

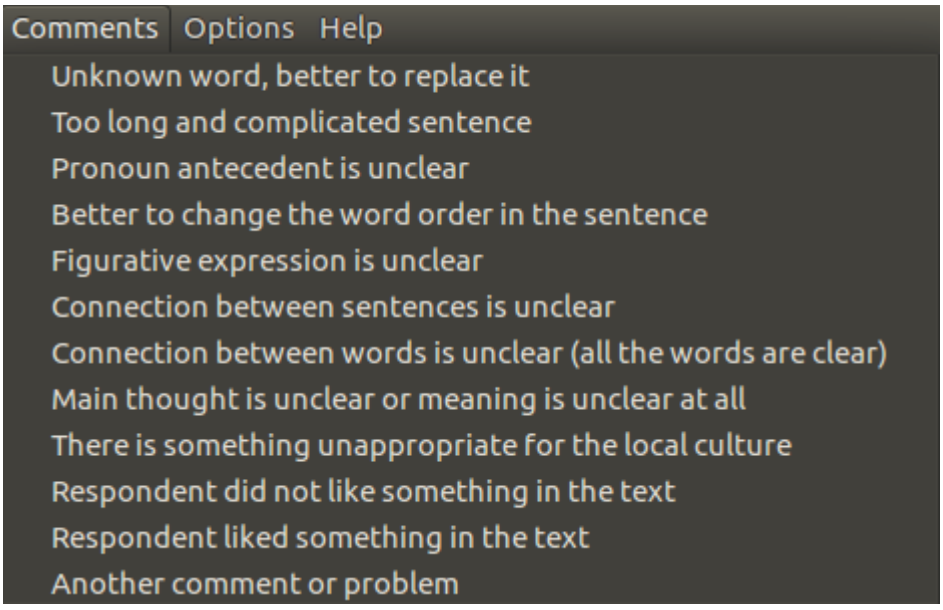

Рис. 10. Подменю “Комментарий”, английский вариант в Linux

7.3. Автоматическая вставка заголовка (позволяет автоматически вставлять заголовок в новый отчет)

7.4. Автоматическая загрузка файлов (по умолчанию программа сохраняет названия открываемых документов и пути к ним и открывает

эти документы при следующем запуске программы; данная опция позволяет отменить их автоматическое открытие при запуске)

7.5. USFM -> HTML (позволяет просматривать HTML код, который получается при преобразовании USFM в HTML)

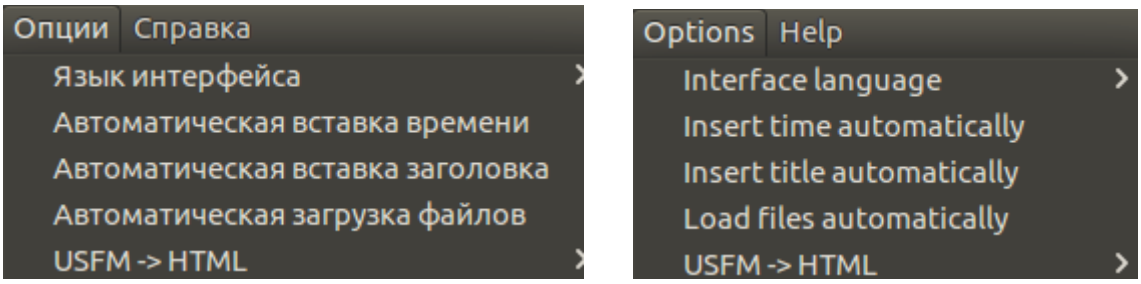

Рис. 11. Подменю “Опции”, русский и английский варианты в Linux

8. Справка

8.1. О программе

Рис. 12. Подменю “Справка”, русский и английский варианты в Linux

3.3. Панель инструментов

На панель инструментов вынесены некоторые наиболее часто используемые опции из главного меню, а также опции для форматирования текста отчета, которых нет в главном меню.

Список опций, доступных на панели инструментов:

- Создать новый файл отчета
- Открыть файл отчета
- Сохранить отчет

- Отменить
- Вернуть
- Вырезать
- Вставить
- Копировать и вставить из файла с вопросами
- Копировать и вставить из файла отчета
- Копировать и вставить из текста перевода
- Копировать и вставить из русского текста
- Шрифт отчета (позволяет изменять шрифт текста отчета)
- Жирный шрифт
- Курсив
- Подчеркивание
- Выравнивание слева
- Выравнивание по центру
- Выравнивание с обеих сторон
- Выравнивание справа
- Цвет шрифта отчета
- Цвет фона текста отчета

Рис. 13. Панель инструментов в Linux

3.4. Контекстные меню

Контекстные меню (доступные по правому клику мышкой внутри одного из текстовых полей) дублируют опции главного меню, но только для соответствующего текстового поля.

Контекстное меню текстового поля для отчета по апробации содержит следующие опции:

- Новый (создание нового файла отчета)
- Открыть
- Сохранить
- Сохранить как
- Закрыть
- Печать
- Предварительный просмотр
- Конвертировать в PDF
- Конвертировать в ODT
- Кодировка
- Отменить
- Вернуть
- Вырезать
- Копировать
- Вставить
- Копировать и вставить
- Удалить
- Выделить всё
- Найти
- Найти и заменить
- Найти и заменить везде
- Увеличить
- Уменьшить

Опции из подменю Ответ респондента и Комментарий недоступны через контекстное меню. Они доступны только через главное меню.

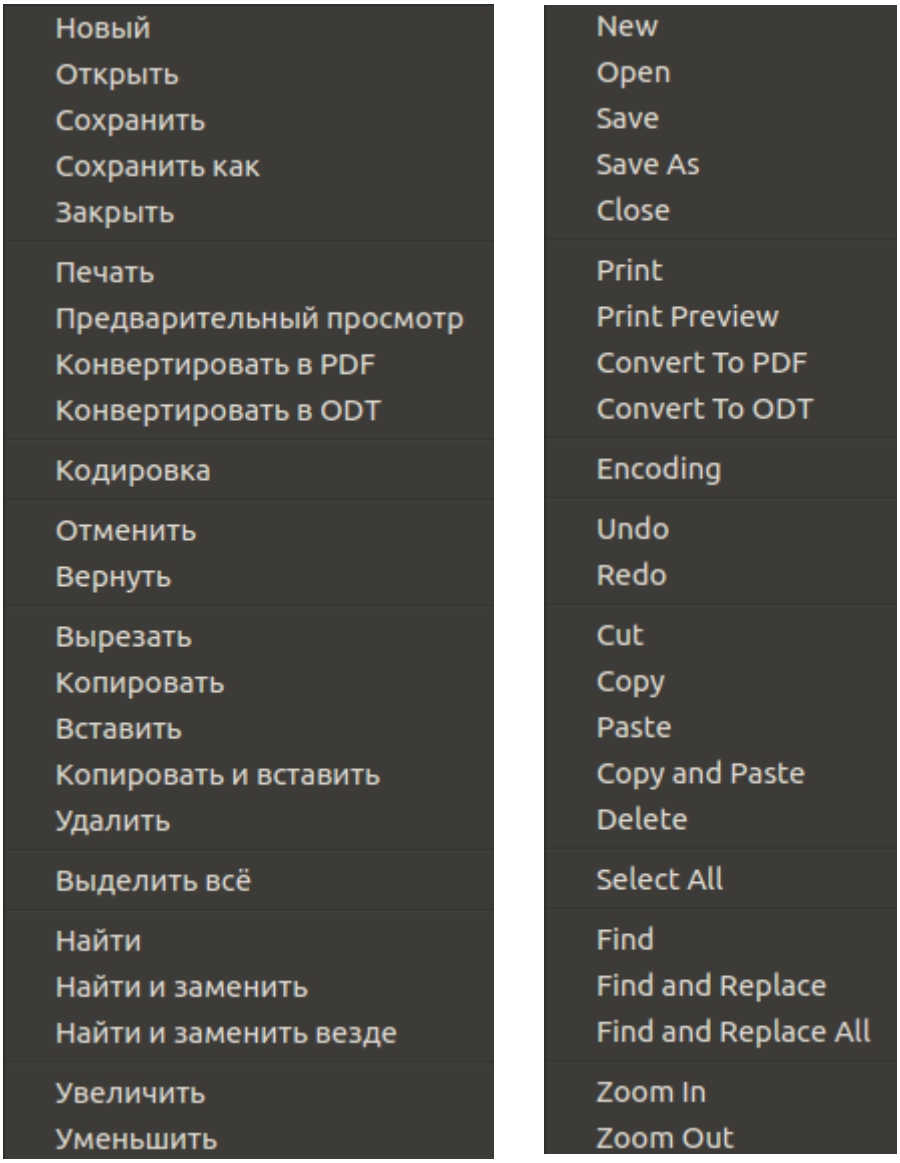

Рис. 14. Контекстное меню для отчета, русский и английский варианты в Linux

Контекстные меню для текста перевода и русского текста отличаются от контекстного меню для отчета. Они содержат следующие опции:

- Открыть
- Закрыть
- Печать
- Предварительный просмотр

- Конвертировать в PDF
- Конвертировать в ODT
- Форматированный (USFM / Paratext)
- Неформатированный (USFM / Paratext)
- Кодировка
- Копировать
- Копировать и вставить в отчет
- Выделить всё
- Найти
- Увеличить
- Уменьшить

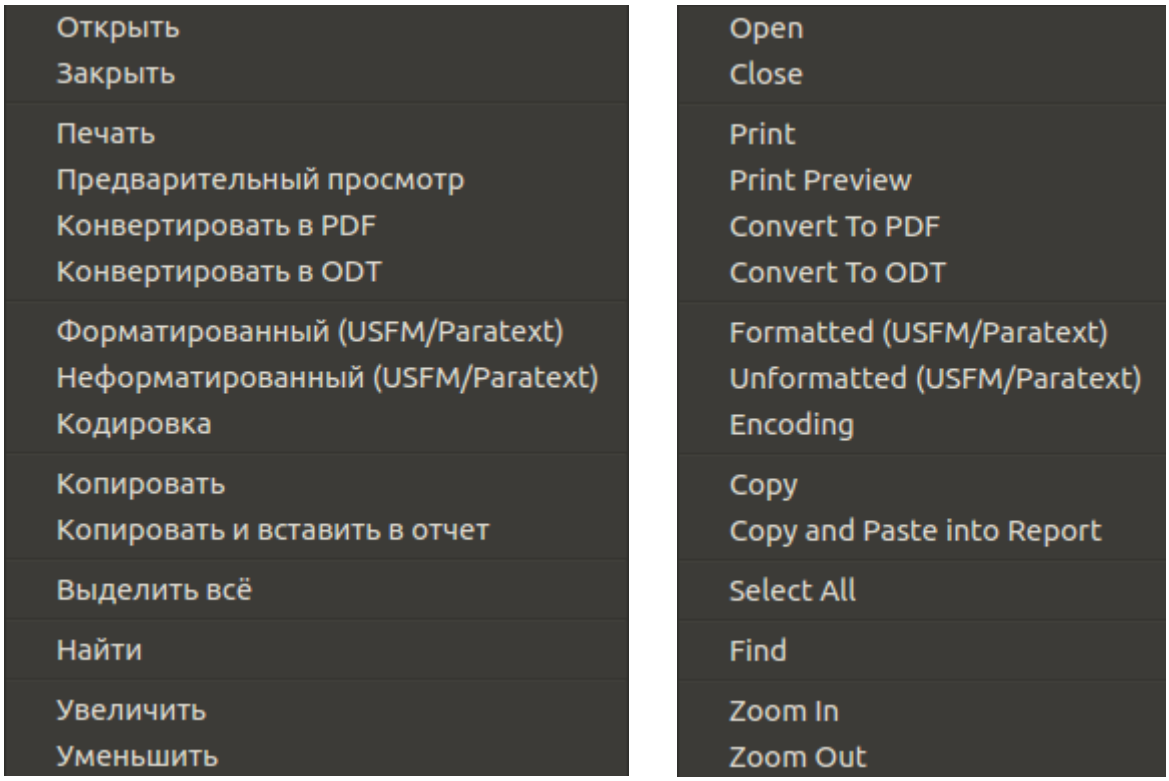

Рис. 15. Контекстное меню для текста перевода и русского текста, русский и английский варианты в Linux

Контекстное меню для текста вопросов имеет еще одну опцию – форматированный (Transcelerator), которая предназначена для форматирования списка вопросов из Transcelerator – плагина для Paratext, предназначенном для создания списков вопросов для апробации.

Хотя Transcelerator обычно генерирует списки вопросов в формате HTML, есть еще один формат списка вопросов Transcelerator. Файлы такого формата имеют расширение .SFM (как и файлы Paratext), однако набор используемых маркеров сильно отличается от маркеров USFM, используемых в Paratext. Для правильного форматирования таких файлов создается специальная опция и конвертер.

В случае использования обычных файлов Transcelerator в формате HTML они также будут представлены в форматированном виде при выборе этой опции. Таким образом, эта опция подходит для любых файлов Transcelerator.

3.5. Внешний вид программы (главного окна)

Qt использует API соответствующей операционной системы, поэтому главное окно программы будет выглядеть по-разному в Linux и Windows, и его внешний вид будет типичным для программ в соответствующей операционной системы. То же касается и меню программы, приведенных выше. В Windows они будут иметь вид, характерный для меню программ на Windows.

Для облегчения ориентации пользователя программы активное в данный момент текстовое поле будет выделено рамкой, а метка с заголовком будет выделяться цветом. При запуске программы по умолчанию активным становится поле отчета, поскольку это наиболее важное поле для пользователя.

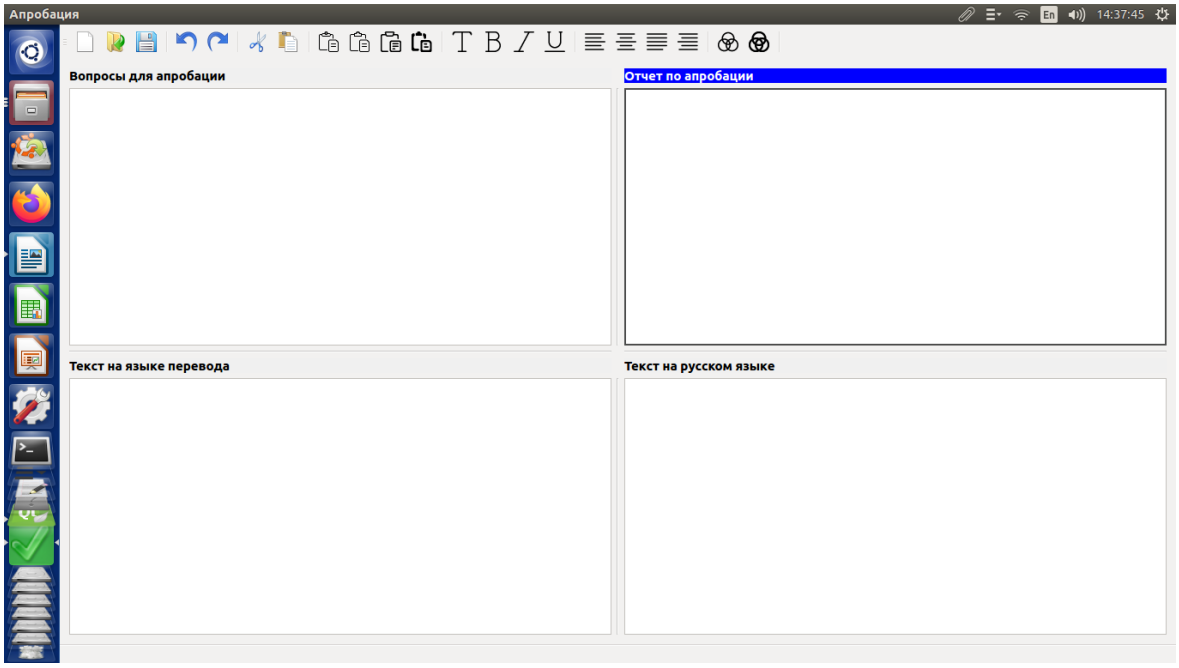

Рис. 16. Внешний вид программы с русским интерфейсом в Linux

Глава 4. Разработка программного кода методов, запускаемых при открытии программы и при ее закрытии

4.1. Разработка главной функции программы

Как обычно, исполнение программы на C++ начинается с главной функции, которая в Qt имеет некоторые особенности.

Поскольку разрабатываемая программа позволяет выбирать язык интерфейса (русский или английский), файл с русским переводом подключается в главной функции. При первом запуске программы язык интерфейса определяется настройками операционной системы, а при последующих запусках – выбором языка, сохраненном в файле настроек (если пользователь изменил язык интерфейса через соответствующую опцию главного меню).

Ниже приведен код главной функции программы.

```
int main(int argc, char *argv[])
{
   QApplication a(argc, argv); //Создается объект приложения
   QTranslator translator; //Создается объект переводчика (который требуется для подключения файла русификации)

   QSettings *m_settings = new QSettings("aprobacija_settings.ini", QSettings::IniFormat); //В файле aprobacija_settings.ini сохраняются настройки программы, в том числе настройки языка
   m_settings->beginGroup("MainWindow"); //Начало чтения файла настроек
   QVariant lang; //В этой переменной хранятся настройки выбранного языка
   QString language = m_settings->value("language", lang).toString();
   m_settings->endGroup(); //Окончание чтения файла настроек

   if (language == "Russian") { //Если в настройках был выбран русский язык, то подключается файл русификации
      translator.load("main_ru.qm"); //Загружается файл русификации
   }
```

```
    else if (language == "") { //Если язык системы русский, то
подключается файл русификации
        QString str = QLocale::system().name(); //Получение сведений о языке
системы
        if ((str == "ru") || (str == "ru_RU") {
            translator.load("main_ru.qm");
        }
    }

    a.installTranslator(&translator); //Объект переводчик с загруженным
файлом русификации (если он был загружен) подключается к
приложению

    MainWindow w; //Создание главного окна программы
    w.show(); //Загрузка главного окна

    return a.exec(); //Запуск приложения
}
```

4.2. Конструктор главного окна

Главное окно загружается главной функцией программы, приведенной выше. Далее приведен код конструктора:

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this); //Создается графический интерфейс

    ZoomReport = ZoomQuestions = ZoomTargetText = ZoomBaseText =
0; //Инициализация этих переменных (чтобы слоты для увеличения и
уменьшения шрифтов в текстовых полях могли корректно
инкрементировать и декрементировать эти переменные, если в файле
настроек они не записаны)

    EncodingReport = EncodingQuestions = EncodingTargetText =
EncodingBaseText = EncodingHTML = "UTF-8";
 //По умолчанию кодировка во всех текстовых полях устанавливается
UTF-8
```

```
    m_settings = new QSettings("aprobacija_settings.ini",
QSettings::IniFormat, this);
    readSettings(); //Запускается функция, считывающая все данные из
файла с настройками (а не только данные об установленном языке)

    if (!FilesNotLoadAuto) {
        open_files_at_start(); //Если не была отменена автоматическая
загрузка файлов, то загружаются тексты из файлов, с которыми
пользователь работал в прошлый раз – каждый в свое текстовое поле
    }

    textEdit_focused("Report"); //Текстовое поле отчета и метка с его
заголовком выделяются, а также устанавливается шрифт во всех метках
    set_zoom_at_start(); //В текстовых окнах устанавливается выбранный
масштаб шрифтов, если эти данные были сохранены в файле настроек

}
```

4.3. Методы для записи и чтения файла настроек

Настройки сохраняются в файле с расширением .ini, поскольку такие файлы хорошо работают и в Linux, и в Windows.

При выходе из программы настройки сохраняются в файле настроек, а при запуске программы – считываются из него.

Для этих методов используются следующие переменные:

```
    QString FilenameReport; //Имя файла с отчетом и путь к нему
    QString FilenameQuestions; //Имя файла с вопросами и путь к нему
    QString FilenameTargetText; //Имя файла с текстом перевода и путь к
нему
    QString FilenameBaseText; //Имя файла с русским текстом и путь к
нему
    QString language; //Выбранный язык
    bool TimeAuto; //Установлена ли автоматическая вставка времени
    bool TitleAuto; //Установлена ли автоматическая вставка заголовка
файла
    bool FilesNotLoadAuto; //Отключена ли автоматическая загрузка
файлов
    int ZoomReport; //Масштаб шрифта отчета
    int ZoomQuestions; //Масштаб шрифта вопросов
```

```
    int ZoomTargetText; //Масштаб шрифта текста перевода
    int ZoomBaseText; //Масштаб шрифта русского текста

    QSettings *m_settings; //Объект настроек; переменные перечисленные ниже соответствуют тем же, что и выше, но объект настроек работает с объектами QVariant
    QVariant report;
    QVariant questions;
    QVariant target_text;
    QVariant base_text;
    QVariant lang;
    QVariant time_auto;
    QVariant title_auto;
    QVariant files_not_load_auto;
    QVariant zoom_report;
    QVariant zoom_questions;
    QVariant zoom_target_text;
    QVariant zoom_base_text;
```

Код метода записи файла настроек:

```
void MainWindow::writeSettings() {

    m_settings->beginGroup("MainWindow");
    m_settings->setValue("pos", pos()); //Позиция главного окна программы на экране
    m_settings->setValue("size", size()); //Размер главного окна на экране

    m_settings->setValue("report", FilenameReport);
    m_settings->setValue("questions", FilenameQuestions);
    m_settings->setValue("target_text", FilenameTargetText);
    m_settings->setValue("base_text", FilenameBaseText);
    m_settings->setValue("language", language);
    m_settings->setValue("time_setting", TimeAuto);
    m_settings->setValue("title_setting", TitleAuto);
    m_settings->setValue("files_not_loading", FilesNotLoadAuto);

    m_settings->setValue("zoom_report", ZoomReport);
    m_settings->setValue("zoom_questions", ZoomQuestions);
    m_settings->setValue("zoom_target_text", ZoomTargetText);
    m_settings->setValue("zoom_base_text", ZoomBaseText);
```

```
    m_settings->endGroup();
    m_settings->sync();
}
```

Код метода чтения файла настроек:

```
void MainWindow::readSettings() {

    QDesktopWidget desk;
    QRect rect = desk.availableGeometry();
    m_settings->beginGroup("MainWindow");
    QPoint pos = m_settings->value("pos", QPoint(0, 0)).toPoint();
    QSize size = m_settings->value("size", rect.size()).toSize();
    if (pos.x() < 0) { //Левый край главного окна программы не должен
выходить за пределы левого края экрана
        pos.setX(0);
    }
    if (pos.y() < 0) { //Верхний край главного окна программы не должен
выходить за пределы верхнего края экрана
        pos.setY(0);
    }
    if (size.width() > rect.width()) { //Ширина главного окна программы не
должна превышать ширину экрана
        size.setWidth(rect.width());
    }
    if (size.height() > rect.height()) { //Высота главного окна программы не
должна превышать высоту экрана
        size.setHeight(rect.height());
    }

    resize(size);
    move(pos);

    FilenameReport = m_settings->value("report", report).toString();
    FilenameQuestions = m_settings->value("questions", questions).toString();
    FilenameTargetText = m_settings->value("target_text",
target_text).toString();
    FilenameBaseText = m_settings->value("base_text", base_text).toString();
    language = m_settings->value("language", lang).toString();
    TimeAuto = m_settings->value("time_setting", time_auto).toBool();
    TitleAuto = m_settings->value("title_setting", title_auto).toBool();
    FilesNotLoadAuto = m_settings->value("files_not_loading",
files_not_load_auto).toBool();
```

```
    ZoomReport = m_settings->value("zoom_report", zoom_report).toInt();
    ZoomQuestions = m_settings->value("zoom_questions",
zoom_questions).toInt();
    ZoomTargetText = m_settings->value("zoom_target_text",
zoom_target_text).toInt();
    ZoomBaseText = m_settings->value("zoom_base_text",
zoom_base_text).toInt();

    m_settings->endGroup();
}
```

4.4. Метод для выделения одного из текстовых полей и метки с заголовком к нему

С помощью этого метода достигается эффект выделения текстового поля, с которым пользователь работает в данный момент.

```
void MainWindow::textEdit_focused(QString textEditName) {

    //выделение выбранного текстового поля и заголовка
    if (textEditName == "Report") {
        ui->textEdit_Report->setFrameStyle(QFrame::WinPanel |
QFrame::Plain);
        ui->label_Report->setStyleSheet("color: rgb(255, 255, 255);background-
color: rgb(0, 0, 255);");
    }
    else if (textEditName == "Questions") {
        ui->textEdit_Questions->setFrameStyle(QFrame::WinPanel |
QFrame::Plain);
        ui->label_Questions->setStyleSheet("color: rgb(255, 255,
255);background-color: rgb(0, 0, 255);");
    }
    else if (textEditName == "TargetText") {
        ui->textEdit_TargetText->setFrameStyle(QFrame::WinPanel |
QFrame::Plain);
        ui->label_TargetText->setStyleSheet("color: rgb(255, 255,
255);background-color: rgb(0, 0, 255);");
    }
    else if (textEditName == "BaseText") {
```

```
        ui->textEdit_BaseText->setFrameStyle(QFrame::WinPanel |
QFrame::Plain);
        ui->label_BaseText->setStyleSheet("color: rgb(255, 255,
255);background-color: rgb(0, 0, 255);");
    }

    //отмена выделения остальных текстовых полей и заголовков
    iif (textEditName != "Report") {
        ui->textEdit_Report->setFrameStyle(QFrame::StyledPanel |
QFrame::Sunken);
        ui->label_Report->setStyleSheet("color: rgb(0, 0, 0);background-color:
rgb(240, 240, 240);");
    }
    if (textEditName != "Questions") {
        ui->textEdit_Questions->setFrameStyle(QFrame::StyledPanel |
QFrame::Sunken);
        ui->label_Questions->setStyleSheet("color: rgb(0, 0, 0);background-
color: rgb(240, 240, 240);");
    }
    if (textEditName != "TargetText") {
        ui->textEdit_TargetText->setFrameStyle(QFrame::StyledPanel |
QFrame::Sunken);
        ui->label_TargetText->setStyleSheet("color: rgb(0, 0, 0);background-
color: rgb(240, 240, 240);");
    }
    if (textEditName != "BaseText") {
        ui->textEdit_BaseText->setFrameStyle(QFrame::StyledPanel |
QFrame::Sunken);
        ui->label_BaseText->setStyleSheet("color: rgb(0, 0, 0);background-
color: rgb(240, 240, 240);");
    }
}
```

4.5. Метод для открытия файлов, с которыми пользователь работал в предыдущий раз

```
void MainWindow::open_files_at_start() {

    //открытие отчета
    if (FilenameReport != "") {
        open_report(FilenameReport); //Вызывается метод открытия отчета
```

```
    }

    else {
        if (TitleAuto) {
            on_actionInsert_report_title_triggered();
        }
    }

    //открытие вопросов
    if (FilenameQuestions != "") {
        open_questions(FilenameQuestions); //Вызывается метод открытия
вопросов
    }

    //открытие текста перевода
    if (FilenameTargetText != "") {
        open_target_text(FilenameTargetText); //Вызывается метод открытия
текста перевода
    }

    //открытие русского текста
    if (FilenameBaseText != "") {
        open_base_text(FilenameBaseText); //Вызывается метод открытия
русского текста
    }

}
```

4.6. Методы для открытия файлов

Разрабатываемая программа позволяет открывать и сохранять файлы трех форматов – 1) обычный текстовый файл (.txt); 2) HTML (который Qt открывает как форматированный текст); 3) USFM (файлы, с которыми работает программа Paratext и которые могут иметь расширение .sfm или .SFM). Файлы первых двух форматов могут быть открыты в любом из текстовых полей. Файлы с расширением .SFM могут быть открыты в текстовом поле для текста перевода или русского текста (по умолчанию – в форматированном виде), а также в текстовом

поле для файла вопросов (по умолчанию – в неформатированном виде, поскольку в этом случае их форматирование может быть разным).

Файлы, открываемые в текстовом поле отчета, открываются для чтения и записи, а файлы, открываемые в остальных текстовых полях, открываются только для чтения.

Код метода открытия файла отчета с небольшими сокращениями (часть кодировок не приведена):

```
void MainWindow::open_report(QString file_report) {
   QFile FileReport(file_report);
   if(FileReport.open(QFile::ReadOnly | QFile::Text))
   {
      QTextStream in(&FileReport);

      if (EncodingReport == "UTF-8") in.setCodec("UTF-8"); //Установка
кодировки
      else if (EncodingReport == "Unicode") in.setCodec("Unicode");
      else if (EncodingReport == "Windows-1251") in.setCodec("Windows-
1251");
      else if (EncodingReport == "KOI8-R") in.setCodec("KOI8-R");
      else if (EncodingReport == "IBM866") in.setCodec("IBM866");
      else if (EncodingReport == "ISO 8859-5") in.setCodec("ISO 8859-5");

……...
      else in.setCodec("UTF-8");

      QString text = in.readAll();
      FileReport.close();

      if (FilenameReport.endsWith(".html") ||
FilenameReport.endsWith(".htm")) {
         ui->textEdit_Report->setAcceptRichText(true);
      }
      else {
         ui->textEdit_Report->setAcceptRichText(false);
      }
      ui->textEdit_Report->setText(text);
      TextReport = text;
```

```
        //перемещение курсора в конец файла (чтобы по умолчанию
изменения в файл отчета вносились в конец файла, а не в его начало)
        if (!(FilenameReport.endsWith(".html") ||
FilenameReport.endsWith(".htm"))) {
            QTextCursor cursor = ui->textEdit_Report->textCursor();
            int pos = text.length();
            cursor.setPosition(pos);
            ui->textEdit_Report->setTextCursor(cursor);
            ui->textEdit_Report->insertPlainText("\n");
            TextReport = text + "\n"; //Эта переменная используется для
проверки, был ли изменен файл отчета
        }
        if (TimeAuto) {
            on_actionInsert_time_of_work_start_triggered();
        }
    }
}
```

Поскольку остальные файлы открывается только для чтения, курсор не требуется смещать в конец файла. Также не требуется проверка, был ли изменен файл, и не требуется вставка начала времени работы.

Кроме того, переменные EncodingReport (кодировка отчета) и FilenameReport (имя файла отчета) заменяются на переменные для соответствующего поля, а название текстового поля textEdit_Report (текстовое поле отчета) – на название соответсвующего текстового поля.

Файл, открываемый в текстовом поле вопросов, открывается без форматирования, поэтому в коде слота вместо проверки расширения файла и соответствующего выбора отображения текста используется одна строка – ui->textEdit_Questions->setText(text).

Файлы, открываемые в текстовых полях для текста перевода и русского текста, открываются с форматированием, но поскольку, в отличие от файла отчета, эти файлы могут быть формата USFM, для них

код проверки условия будет другим. Для метода открытия текста перевода он выглядит так:

```
    if (FilenameTargetText.endsWith(".sfm") ||
FilenameTargetText.endsWith(".SFM")) {
        QString text_converted = converter_to_html(text);
        ui->textEdit_TargetText->setHtml(text_converted);
    }
    else {
        ui->textEdit_TargetText->setText(text);
    }
```

4.7. Метод для установки масштаба шрифта в текстовых полях при запуске программы

```
void MainWindow::set_zoom_at_start() {

  if (ZoomReport > 0) {
    for (int n = 0; n < ZoomReport; n++) {
      ui->textEdit_Report->zoomIn();
    }
  }
  else if (ZoomReport < 0) {
    for (int n = 0; n > ZoomReport; n--) {
      ui->textEdit_Report->zoomOut();
    }
  }

  if (ZoomQuestions > 0) {
    for (int n = 0; n < ZoomQuestions; n++) {
      ui->textEdit_Questions->zoomIn();
    }
  }
  else if (ZoomQuestions < 0) {
    for (int n = 0; n > ZoomQuestions; n--) {
      ui->textEdit_Questions->zoomOut();
    }
  }

  if (ZoomTargetText > 0) {
    for (int n = 0; n < ZoomTargetText; n++) {
```

```
            ui->textEdit_TargetText->zoomIn();
        }
    }
    else if (ZoomTargetText < 0) {
        for (int n = 0; n > ZoomTargetText; n--) {
            ui->textEdit_TargetText->zoomOut();
        }
    }

    if (ZoomBaseText > 0) {
        for (int n = 0; n < ZoomBaseText; n++) {
            ui->textEdit_BaseText->zoomIn();
        }
    }
    else if (ZoomBaseText < 0) {
        for (int n = 0; n > ZoomBaseText; n--) {
            ui->textEdit_BaseText->zoomOut();
        }
    }
}
```

4.8. Метод закрытия программы

Код этого метода:

```
  void MainWindow::closeEvent(QCloseEvent *event) {
    close_report(); //Вызывается метод закрытия файла отчета
    QMessageBox* messagebox = new QMessageBox
(QmessageBox::Question, tr("Confirm"), tr("Exit program?"),
QMessageBox::Yes | QMessageBox::No, this); //Запрос подтверждения
выхода из программы
    messagebox->setButtonText(QMessageBox::Yes, tr("Yes"));
    messagebox->setButtonText(QMessageBox::No, tr("No"));
    int n = messagebox->exec();
    delete messagebox;

    if (n == QMessageBox::Yes) {
        writeSettings(); //Запись в файл настроек
        event->accept();
    } else {
        event->ignore();
    }
}
```

4.9. Метод закрытия файла отчета

Этот метод проверяет, был ли изменен файл отчета, перед его закрытием. Если файл был изменен, то выдается запрос на его сохранение. Кроме того, перед закрытием файла отчета вставляется время окончания работы, если была установлена автоматическая вставка времени.

```
void MainWindow::close_report() {
QString str = ui->textEdit_Report->toPlainText();
   if ((!(str.isEmpty() || str.isNull())) && TimeAuto) { //Если была установлена автоматическая вставка времени, то в текст отчета вставляется время завершения работы
       on_actionInsert_time_of_work_end_triggered();
       str = str + "Work finished"; //Эти слова добавляются к проверочной строке, чтобы автоматически вставленное время окончания работы было сохранено
   }
   if ((!(str.isEmpty() || str.isNull())) && (str != TextReport)) { //Если был изменен текст отчета, вызывается сообщение с запросом, нужно ли сохранять изменения
       QMessageBox* messagebox = new QMessageBox (QmessageBox::Warning, tr("Warning"), tr("Do you want to save the report file?"), QMessageBox::Yes | QMessageBox::No, this);
       messagebox->setButtonText(QMessageBox::Yes, tr("Yes"));
       messagebox->setButtonText(QMessageBox::No, tr("No"));
       int n = messagebox->exec();
       delete messagebox;
       if (n == QMessageBox::Yes) {
          if (FilenameReport.isEmpty() || FilenameReport.isNull()) { //Если файл отчета еще не был сохранен ранее, вызывается слот “Сохранить как”, позволяющий выбрать имя для сохраняемого файла
             on_actionSave_report_as_triggered();
          }
          else { //Если файл отчета ранее был сохранен, то он будет сохранен под тем же именем
             on_actionSave_report_triggered();
          }
```

```
        }
    }
}
```

4.10. Слоты выбора текстовых полей

Эти слоты соединены с сигналами выбора соответствующего текстового поля и вызывают метод для выделения одного из текстовых полей и метки с заголовком к нему (который был рассмотрен ранее). Этих слотов четыре, по количеству текстовых полей. Код слота выбора текстового поля отчета:

```
void MainWindow::on_textEdit_Report_selectionChanged()
{
    textEdit_focused("Report");
}
```

Слоты для остальных текстовых полей отличаются только строкой, которая передается в метод textEdit_focused().

Глава 5. Разработка программного кода для обработки опций главного меню

В терминологии, используемой в Qt, события (нажатие клавиши, выбор опции меню и т.д.) называются сигналами, а обработчики событий – слотами. Qt Creator предоставляет удобные опции для связи сигналов главного меню и панели инструментов со слотами без необходимости писать код для связи сигналов и слотов.

5.1. Слоты для обработки сигналов подменю “Файл”

5.1.1. Слоты опций “Открыть”

В главном меню есть четыре опции для открытия файлов – для открытия каждого из четырех документов: файла с вопросами, файлв отчета, текста перевода и русского текста.

Код слотов для этих опций отличается незначительно. Для примера ниже приведен код слота для опции “Открыть файл с вопросами”.

```
void MainWindow::on_actionOpen_questions_triggered()
{
   textEdit_focused("Questions");
   QString file = QFileDialog::getOpenFileName(this, tr("Open the questions
file"), "", tr("All files for opening (*.txt *.sfm *.SFM *.html *.htm);;\
Text files (*.txt);;Paratext files (*.sfm *.SFM);;HTML files (*.html *.htm)"));

   if(!file.isEmpty()) {
      FilenameQuestions = file;
      open_questions(FilenameQuestions);
   }
}
```

Слоты для открытия файлов в других текстовых полях отличаются строкой, которая передается в метод textEdit_focused(), вызываемым

методом для открытия файла (которые были рассмотрены в п. 4.6), а также переменной с именем открываемого файла, которая передается в этот метод.

Слот для опции “Открыть файл отчета” также отличается тем, что перед вызовом диалога открытия файла сначала вызывается метод закрытия файла (который был открыт ранее) close_report(). Этот метод в свою очередь вызывает метод сохранения файла и таким образом содержимое файла отчета не будет потеряно, если оно не было сохранено перед выбором опции “Открыть файл отчета”.

Кроме того, слот опции “Открыть файл отчета” позволяет открывать только файлы формата HTML и TXT, но не файлы формата USFM (файлы Paratext).

5.1.2. Слоты опций “Закрыть”

В эту группу также входят четыре опции для закрытия соответствующих файлов. Фактически, файлы закрываются опциями “Открыть” сразу после считывания данных из них, а опции “Закрыть” просто удаляют текст из текстового поля. Код слота опции “Закрыть файл отчета”:

```
void MainWindow::on_actionClose_report_triggered()
{
   textEdit_focused("Report");
   close_report(); //В случае файла отчета нужно убедиться, что все
изменения сохранены, поэтому перед удалением данных из текстового
поля вызывается специальный метод
   ui->textEdit_Report->setText("");
}
```

Для остальных файлов вызов метода close_report() не требуется. Кроме того, они отличаются текстовым полем, из которого удаляется текст, а также строкой, передаваемой в метод textEdit_focused().

5.1.3. Слот опции “Создать новый файл отчета”

Код слота для этой опции:

```
void MainWindow::on_actionNew_report_triggered() {
   textEdit_focused("Report");
   close_report(); //Закрывается ранее открытый файл отчета
   FilenameReport = "";
   ui->textEdit_Report->setAcceptRichText(true);
   ui->textEdit_Report->setText("");
   if (TitleAuto) {
       on_actionInsert_report_title_triggered();
   }
   if (TimeAuto) {
      on_actionInsert_time_of_work_start_triggered();
   }
}
```

5.1.4. Слот опции “Сохранить отчет”

Отчет сохраняется в уже существующем файле отчета (файл перезаписывается).

```
void MainWindow::on_actionSave_report_triggered()
{
   textEdit_focused("Report");
   QFile FileReport(FilenameReport);
   if (FilenameReport.isEmpty() || FilenameReport.isNull()) {
      on_actionSave_report_as_triggered(); //Если имя файла отчета
отсутствует, то требуется вызвать диалог для выбора имени и пути
сохранения файла. Для этого вызывается метод опции “Сохранить отчет
как”.
   }

   if(FileReport.open(QFile::WriteOnly | QFile::Text))
   {
      QTextStream out(&FileReport);

      if (EncodingReport == "UTF-8") out.setCodec("UTF-8");
```

```
        else if (EncodingReport == "Unicode") out.setCodec("Unicode");
        else if (EncodingReport == "Windows-1251") out.setCodec("Windows-
1251");
    ……...
        else out.setCodec("UTF-8");
        if (ui->textEdit_Report->acceptRichText()) {
           out << ui->textEdit_Report->toHtml();
        }
        else {
           out << ui->textEdit_Report->toPlainText();
        }
        FileReport.flush();
        FileReport.close();

        TextReport = ui->textEdit_Report->toPlainText();
    }
}
```

5.1.5. Слот опции “Сохранить отчет как”

Слот для этой опции предоставляет возможность выбрать имя файла и место его сохранения, а после этого для сохранения файла вызывается слот для опции “Сохранить отчет”.

```
void MainWindow::on_actionSave_report_as_triggered()
{
    textEdit_focused("Report");
    QStringList items;
    items << tr("Rich text (HTML)") << tr("Plain text (txt)");
    bool ok;
    QString item = QInputDialog::getItem(this, tr("Saving file"), tr("Choose
how to save file:"), items, 0, false, &ok); //Этот диалог позволяет выбрать,
сохранить ли файл отчета как HTML файл (при этом сохраняется
форматирование) или как простой текст без форматирования
    if (ok && !item.isEmpty()) {

        QString file;

        if (item == tr("Rich text (HTML)")) {
            ui->textEdit_Report->setAcceptRichText(true);
```

```
            file = QFileDialog::getSaveFileName(this, tr("Save the report file"),
"", tr("HTML files (*.html)")) + ".html"; //К файлу HTML автоматически
добавляется расширение .html
        }
        else {
            ui->textEdit_Report->setAcceptRichText(false);
            file = QFileDialog::getSaveFileName(this, tr("Save the report file"),
"", tr("Text files (*.txt)")) + ".txt"; //К текстовому файлу автоматически
добавляется расширение .txt
        }

        if(!file.isEmpty())
        {
            QDir direct;
            QString dir = direct.filePath(file);
            FilenameReport = dir;
            on_actionSave_report_triggered();
        }
    }
}
```

5.1.6. Слоты опций “Печать”

В эту группу опций входит четыре опции. Каждая из них служит для вывода на печать соответствующего документа. Код слота опции “Печать файла с вопросами”:

```
void MainWindow::on_actionPrint_questions_triggered()
{
   textEdit_focused("Questions");
   QPrinter printer (QPrinter::HighResolution);
   printer.setPrinterName(tr("Printer name"));
   QPrintDialog dialog(&printer, this);
   if(dialog.exec() == QDialog::Rejected) return;
   ui->textEdit_Questions->print(&printer);
}
```

Остальные слоты отличаются только текстовым полем, файл которого выводится на печать, а также строкой, передаваемой в метод textEdit_focused().

5.1.7. Слоты опций “Предварительный просмотр”

В эту группу опций также входит четыре опции, которые позволяют просматривать соответствующий документ перед выводом на печать. В отличие от опций печати, для предварительного просмотра требуется создать специальный слот предварительного просмотра, связанный с диалогом предварительного просмотра. Поэтому в каждом случае требуется два слота, а не один.

Код для слотов опции “Предварительный просмотр вопросов”:

```
void MainWindow::on_actionPrint_preview_questions_triggered()
{
   textEdit_focused("Questions");
   QPrinter printer(QPrinter::HighResolution);
   QPrintPreviewDialog preview(&printer, this, Qt::Window);
   connect(&preview, SIGNAL(paintRequested(QPrinter*)),
SLOT(paintPreviewQuestions(QPrinter*)));
   preview.exec();
}

void MainWindow::paintPreviewQuestions(QPrinter *printer) {
   ui->textEdit_Questions->print(printer);
}
```

Слоты для опций предварительного просмотра в других текстовых полях отличаются строкой, передаваемой в метод textEdit_focused(), названием слота для предварительного просмотра и текстовым полем в этом слоте.

5.1.8. Опции “Конвертировать в PDF” и “Конвертировать в ODT”

В эту группу входят методы для конвертирования в PDF и в ODT, который осуществляет само конвертирование, а также по четыре опции для конвертирования в PDF и в ODT.

5.1.8.1. Методы для конвертирования в PDF и в ODT

Код метода конвертирования в PDF:

```
void MainWindow::convert_to_PDF(QString writing, QString
document_name) {

    QTextDocument document;
    document.setHtml(writing);

    QString Filename;
    if (document_name == "Questions") {
        Filename = FilenameQuestions;
    }
    else if (document_name == "Report") {
        Filename = FilenameReport;
    }
    else if (document_name == "TargetText") {
        Filename = FilenameTargetText;
    }
    else if (document_name == "BaseText") {
        Filename = FilenameBaseText;
    }
    else return;

    if (Filename.endsWith(".txt")) { //Если название файла имеет
расширение .txt, то надо его удалить и заменить на .pdf
        Filename = Filename.replace(".txt", ".pdf");
    }
    else if (Filename.endsWith(".html")) { //Аналогично для расширения
.html
        Filename = Filename.replace(".html", ".pdf");
    }
    else if (Filename.endsWith(".htm")) {
        Filename = Filename.replace(".htm", ".pdf");
    }
    else if (Filename.endsWith(".sfm")) {
        Filename = Filename.replace(".sfm", ".pdf");
    }
    else if (Filename.endsWith(".SFM")) {
        Filename = Filename.replace(".SFM", ".pdf");
    }
```

```
    else {
        Filename = Filename + ".pdf"; //В остальных случаях, скорее всего, файл вообще не имеет расширения, поэтому добавляем .pdf
    }

    QPrinter printer(QPrinter::HighResolution);
    printer.setOutputFileName(Filename);
    printer.setPaperSize(QPrinter::A4);
    printer.setOutputFormat(QPrinter::PdfFormat);
    document.print(&printer);
    QMessageBox::information(this, tr("Conversion to PDF"), tr("Created file") + " " + Filename);
}
```

Код метода конвертирования в ODT отличается последней частью. Вместо кода для объекта QPrinter этот метод имеет код для объекта QTextDocumentWriter:

```
QTextDocumentWriter writer;
    writer.setFormat("odf");
    writer.setFileName(Filename);
    writer.write(&document);
    QMessageBox::information(this, tr("Conversion to ODT"), tr("Created file") + " " + Filename);
```

5.1.8.2. Слоты опций конвертирования в PDF / ODT”

Код слота опции “Конвертировать вопросы в PDF”:

```
void MainWindow::on_actionConvert_questions_to_PDF_triggered()
{
    textEdit_focused("Questions");
    QString writing = ui->textEdit_Questions->toHtml();
    QString writing_2 = ui->textEdit_Questions->toPlainText();

    if (writing_2.isEmpty() || writing_2.isNull()) { //Проверка, есть ли текст в текстовом поле для вопросов
        QMessageBox::warning(this, tr("Warning"), tr("The quesions file is not open"));
        return;
    }
```

```
    convert_to_PDF(writing, "Questions");
}
```

Слоты для опций “Конвертировать в ODT” отличаются тем, что вместо метода convert_to_PDF() вызывается метод convert_to_ODT().

Слоты для опций конвертирования файлов из других текстовых полей отличаются, во-первых, строкой, передаваемой в метод textEdit_focused() и convert_to_PDF() или convert_to_ODT(), а во-вторых, текстовым полем.

Кроме того, слот для опции “Конвертировать отчет” перед вызовом метода convert_to_PDF() или convert_to_ODT() содержит дополнительный код для проверки имени файла:

```
    if (FilenameReport.isEmpty() || FilenameReport.isNull()) { //Проверка, был ли ранее сохранен файл отчета и соответственно есть ли у него имя
        bool bOK;
        FilenameReport = QInputDialog::getText(this, tr("Choose file name"), tr("Enter file name:"), QLineEdit::Normal, "", &bOK);
        if (!bOK) return;
    }
```

5.1.9. Слот опции “Выйти из программы”

Как следует из названия, эта опция служит для выхода из программы. Слот этой опции вызывает встроенную функцию close(), которая генерирует событие закрытия программы, а для его обработки предусмотрен специальный метод – closeEvent().

```
void MainWindow::on_actionExit_triggered()
{
    textEdit_focused("Report");
    close(); //Этот вызов метода close() вызывает также событие закрытия программы
}
```

5.2. Слоты для обработки сигналов подменю “Редактировать”

5.2.1. Слоты опций “Отменить”, “Повторить”, “Вырезать” и “Вставить”

Опция “Отменить” отменяет предыдущее действие в текстовом поле отчета, а опция “Повторить” повторяет отмененное действие в том же текстовом поле. Код слота для опции “Отменить”:

```
void MainWindow::on_actionUndo_triggered()
{
   textEdit_focused("Report");
   ui->textEdit_Report->undo();
}
```

В коде слота для опции метод undo(), соответственно, заменяется на метод redo().

Опции “Вырезать” и “Вставить” используются соответственно для вырезания выделенного текста из отчета и для вставки в текстовое поле текста из буфера.

Их код аналогичен коду для опций “Отменить” и “Повторить” и отличается только методами в последней строке – cut() для опции “Вырезать” и paste() для опции “Вставить”.

5.2.2. Слоты опций “Копировать”

Эти опции служат для копирования выделенного текста из соответствующего текстового поля. Код слота для опции “Копировать из текста вопросов:

```
void MainWindow::on_actionCopy_from_questions_triggered()
{
   textEdit_focused("Questions");
   ui->textEdit_Questions->copy();
}
```

Остальные слоты отличаются только строкой, передаваемой в метод textEdit_focused(), а также текстовым полем.

5.2.3. Слоты опций “Копировать и вставить”

Опции этой группы служат для ускорения копирования и вставки. Они позволяют совершать копирование и вставку как одну операцию. Сначала выделенный текст копируется из соответствующего текстового поля, а потом вставляется в текстовое поле отчета. Помимо главного меню все опции этой группы доступны на панели инструментов и контекстных меню.

Ниже приведен код слота для опции “Копировать и вставить из файла с вопросами”.

```
void MainWindow::on_actionCopy_and_paste_from_questions_triggered()
{
   textEdit_focused("Report");
   ui->textEdit_Questions->copy();
   ui->textEdit_Report->paste();
   ui->textEdit_Report->insertPlainText("\n");
}
```

Остальные слоты отличаются только текстовым полем, из которого производится копирование:, а также строкой, передаваемой в метод textEdit_focused().

5.2.4. Слот опции “Удалить”

Эта опция удаляет выделенный текст из текстового поля отчета.

```
void MainWindow::on_actionDelete_triggered()
{
   textEdit_focused("Report");
   QTextCursor cursor = ui->textEdit_Report->textCursor();
   cursor.removeSelectedText();
```

```
}
```

5.2.5. Слоты опций “Выделить всё”

Эти опции предназначены для выделения всего текста в соответствующем текстовом поле. Код слота опции “Выделить всё в файле вопросов”:

```
void MainWindow::on_actionSelect_all_from_questions_triggered()
{
   textEdit_focused("Questions");
   ui->textEdit_Questions->selectAll();
}
```

Остальные слоты отличаются только строкой, передаваемой в метод textEdit_focused(), а также текстовым полем.

5.2.5. Опции “Найти”

Для реализации возможности поиска было создано новое диалоговое окно вместе с классом DialogFind, а также метод поиска в классе MainWindow и четыре слота, соответствующих четырем текстовым полям.

5.2.5.1. Диалоговое окно “Поиск”

Оно позволяет использовать несколько опций для поиска – 1) направление поиска (вперед или назад), 2) учитывать регистр, 3) искать слово целиком.

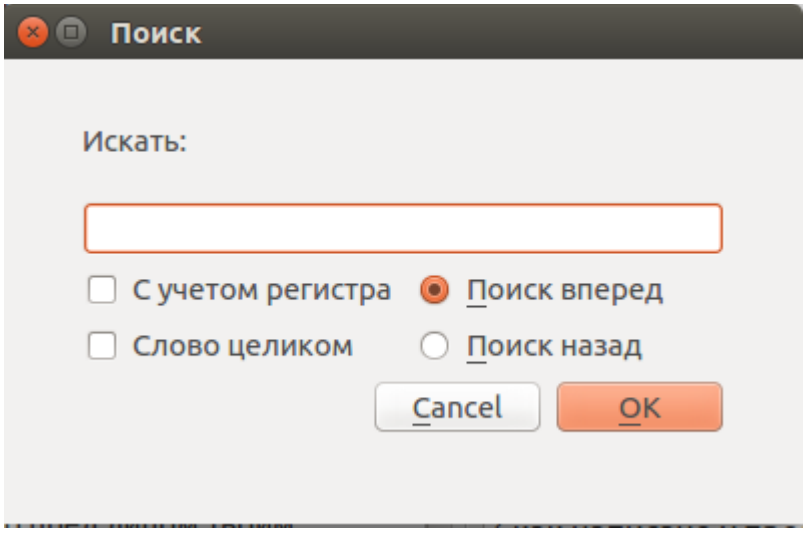

Рис. 17. Диалоговое окно “Поиск”, русский вариант в Linux

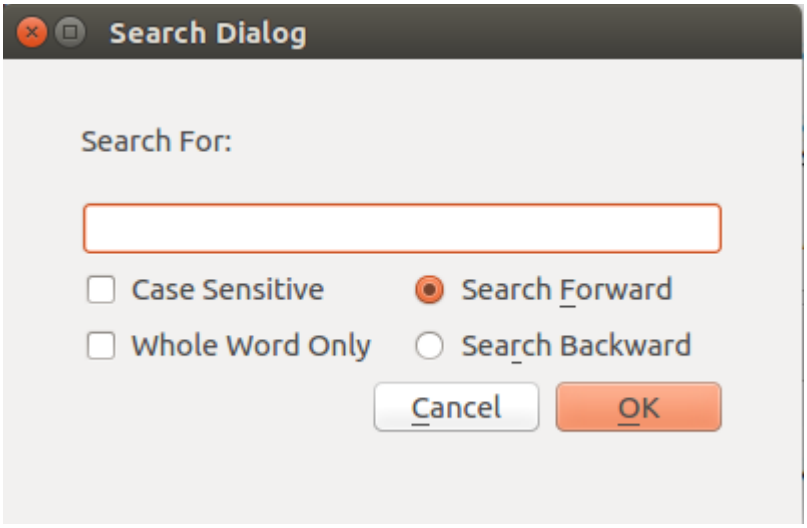

Рис. 18. Диалоговое окно “Поиск”, английский вариант в Linux

5.2.5.2. Класс DialogFind

Этот класс содержит несколько простых слотов:

```
void DialogFind::on_lineEdit_Find_editingFinished()
{
   search = ui->lineEdit_Find->text();
}
void DialogFind::on_checkBox_Case_clicked(bool checked)
```

```
{
   if (checked) {
      CaseSensitive = true;
   }
   else {
      CaseSensitive = false;
   }
}
void DialogFind::on_checkBox_Whole_clicked(bool checked)
{
   if (checked) {
      WholeWord = true;
   }
   else {
      WholeWord = false;
   }
}
void DialogFind::on_radioButton_Forward_clicked()
{
   SearchDirection = "Forward";
}
void DialogFind::on_radioButton_Backward_clicked()
{
   SearchDirection = "Backward";
}
void DialogFind::on_buttonBox_accepted()
{
   BeginSearch = true;
}
void DialogFind::on_buttonBox_rejected()
{
   BeginSearch = false;
}
```

5.2.5.3. Метод для поиска

Этот метод используется всеми четырьмя слотами, связанными с опциями поиска.

```
void MainWindow::search_word(QString document_name) {
```

```
    DialogFind dialogfindDocument;
    dialogfindDocument.exec();
    dialogfindDocument.show(); //Для сохранения показа диалога при
продолжении поиска

    bool Found = false;

    QFlags<QTextDocument::FindFlag> searchOptions;

    if (dialogfindDocument.CaseSensitive == true) {
        if (dialogfindDocument.WholeWord == true) {
            if (dialogfindDocument.SearchDirection == "Forward") {
                searchOptions = QTextDocument::FindCaseSensitively |
QTextDocument::FindWholeWords;
            }
            else if (dialogfindDocument.SearchDirection == "Backward") {
                searchOptions = QTextDocument::FindBackward |
QTextDocument::FindCaseSensitively |
                        QTextDocument::FindWholeWords;
            }
        }
        else {
            if (dialogfindDocument.SearchDirection == "Forward") {
                searchOptions = QTextDocument::FindCaseSensitively;
            }
            else if (dialogfindDocument.SearchDirection == "Backward") {
                searchOptions = QTextDocument::FindBackward |
QTextDocument::FindCaseSensitively;
            }
        }
    }

    else {
        if (dialogfindDocument.WholeWord == true) {
            if (dialogfindDocument.SearchDirection == "Forward") {
                searchOptions = QTextDocument::FindWholeWords;
            }
            else if (dialogfindDocument.SearchDirection == "Backward") {
                searchOptions = QTextDocument::FindBackward |
QTextDocument::FindWholeWords;
            }
```

```
        }
        else {
            if (dialogfindDocument.SearchDirection == "Forward") {
                searchOptions = 0;
            }
            else if (dialogfindDocument.SearchDirection == "Backward") {
                searchOptions = QTextDocument::FindBackward;
            }
        }
    }

    if (document_name == "Questions") {
        Found = ui->textEdit_Questions->find(dialogfindDocument.search,
searchOptions);
    }
    else if (document_name == "Report") {
        Found = ui->textEdit_Report->find(dialogfindDocument.search,
searchOptions);
    }
    else if (document_name == "TargetText") {
        Found = ui->textEdit_TargetText->find(dialogfindDocument.search,
searchOptions);
    }
    else if (document_name == "BaseText") {
        Found = ui->textEdit_BaseText->find(dialogfindDocument.search,
searchOptions);
    }
    else return;

    if ((!(dialogfindDocument.search.isEmpty() ||
dialogfindDocument.search.isNull())) & !Found) {
        QMessageBox* notFound = new
QMessageBox(QMessageBox::Warning, tr("Not found"),
                                    tr("The word was not found"));
        notFound->exec();
        delete notFound;
    }

    if ((!(dialogfindDocument.search.isEmpty() ||
dialogfindDocument.search.isNull())) & Found) {
        bool continueSearch = true;
        bool foundNew = true;
```

```
        while (continueSearch & foundNew) {
            QMessageBox* findNext =
                    new QMessageBox(QMessageBox::Question, tr("Continue?"),
tr("Find next?"), QMessageBox::Yes | QMessageBox::No |
QMessageBox::Cancel);
            findNext->setButtonText(QMessageBox::Yes, tr("Yes"));
            findNext->setButtonText(QMessageBox::No, tr("No"));
            findNext->setButtonText(QMessageBox::Cancel, tr("Cancel"));

            int n = findNext->exec();
            delete findNext;

            if (n == QMessageBox::Yes) {
                if (document_name == "Questions") {
                    foundNew = ui->textEdit_Questions-
>find(dialogfindDocument.search, searchOptions);
                }
                else if (document_name == "Report") {
                    foundNew = ui->textEdit_Report-
>find(dialogfindDocument.search, searchOptions);
                }
                else if (document_name == "TargetText") {
                    foundNew = ui->textEdit_TargetText-
>find(dialogfindDocument.search, searchOptions);
                }
                else if (document_name == "BaseText") {
                    foundNew = ui->textEdit_BaseText-
>find(dialogfindDocument.search, searchOptions);
                }
                else return;
            }
            else if ((n == QMessageBox::No) || (n == QMessageBox::Cancel)) {
                continueSearch = false;
            }
        }

        if (!foundNew) {
            QMessageBox* noNewFound = new
QMessageBox(QMessageBox::Warning, tr("End of search"),
                                        tr("End of search is reached"));
            noNewFound->exec();
            delete noNewFound;
```

```
        }
    }
}
```

5.2.5.4. Слоты опций поиска

Код слота опции “Найти в файле с вопросами:

```
void MainWindow::on_actionFind_in_questions_triggered() {
    textEdit_focused("Questions");
    QString str = ui->textEdit_Questions->toPlainText();
    if (str.isEmpty() || str.isNull()) {
        QMessageBox* warning = new QMessageBox(QMessageBox::Warning,
tr("Warning"), tr("File is not open"));
        warning->exec();
        delete warning;
        return;
    }
    search_word("Questions");
}
```

Слоты опций “Найти в файле отчета”, “Найти в тексте перевода” и “Найти в русском тексте” отличаются строкой, которая передается в методы textEdit_focused() и search_word(), а также текстовым полем, в котором ищется слово.

5.2.6. Опция “Найти и заменить”

Для реализации опции поиска и замены было создано еше одно новое диалоговое окно вместе с классом DialogReplace. Поскольку эта опция используется только для отчета, метод поиска и замены не был вынесен отдельно, а включен в слот для обработки этой опции.

5.2.6.1. Диалоговое окно “Поиск и замена”

Это диалоговое окно включает те же элементы и поисковые опции, что и окно “Поиск”, а также дополнительное текстовое поле для ввода

слова (или фразы), на которое будет заменяться слово (или фраза) для поиска.

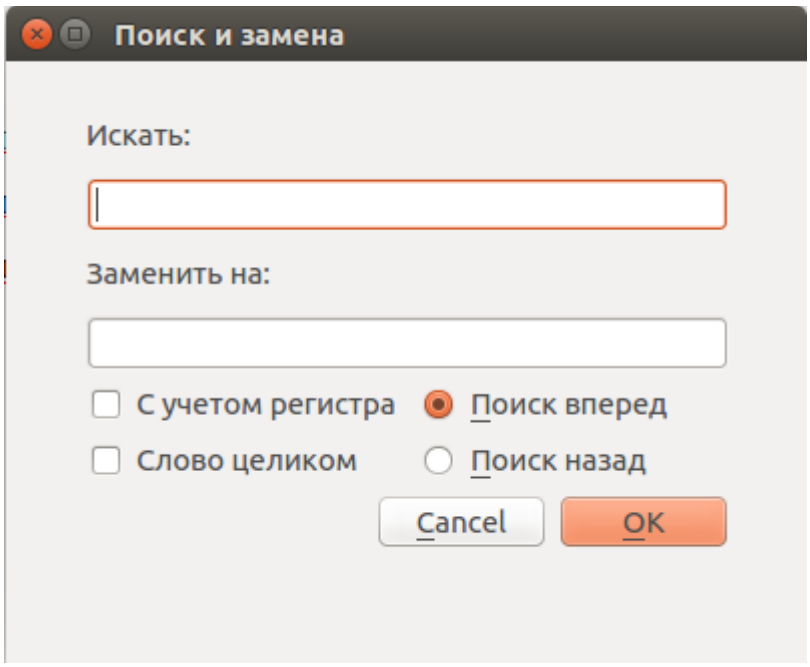

Рис. 19. Диалоговое окно “Поиск и замена”, русский вариант в Linux

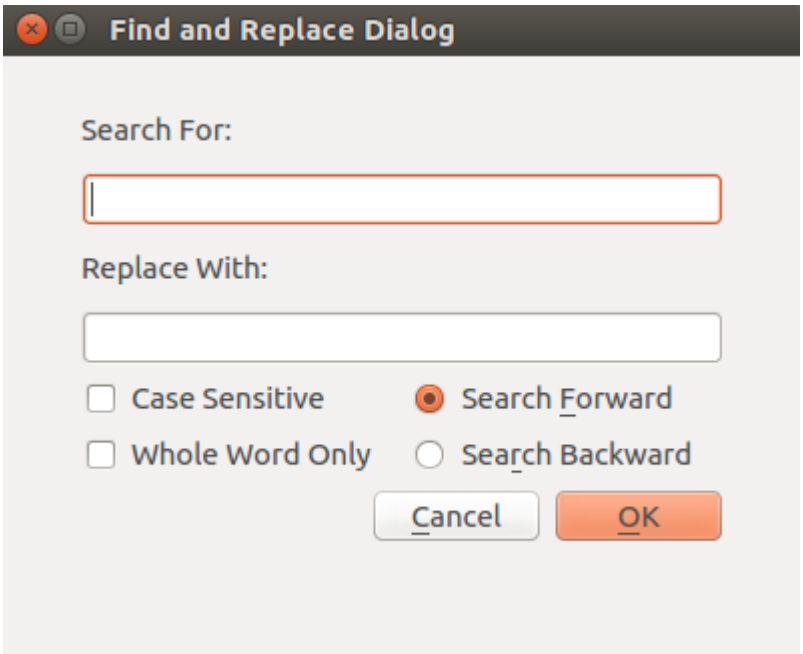

Рис. 20. Диалоговое окно “Поиск и замена”, английский вариант в Linux

5.2.6.2. Класс DialogReplace

По сравнению с классом DialogFind он содержит один дополнительный слот:

```
void DialogReplace::on_lineEdit_Replace_editingFinished()
{
   replace = ui->lineEdit_Replace->text();
}
```

5.2.6.3. Слот опции “Найти и заменить”

Слот для этой опции не разделяется на метод и сам слот, поскольку эта опция есть только для файла отчета.

В отличие от опции “Найти”, слот этой опции также использует методы cut() и insertPlainText():

```
……...
    else if (!(dialogreplace.search.isEmpty() || dialogreplace.search.isNull())) {
       ui->textEdit_Report->cut();
       ui->textEdit_Report->insertPlainText(dialogreplace.replace);
    }
…….
   if (foundNew) {
       ui->textEdit_Report->cut();
       ui->textEdit_Report->insertPlainText(dialogreplace.replace);
   }
…….
```

5.2.7. Опция “Найти и заменить везде”

Для реализации этой опции было создано еше одно новое диалоговое окно вместе с классом DialogReplaceAll. Поскольку эта

опция используется только для отчета, метод поиска и замены не был вынесен отдельно, а включен в слот для обработки этой опции.

5.2.7.1. Диалоговое окно “Найти и заменить везде”

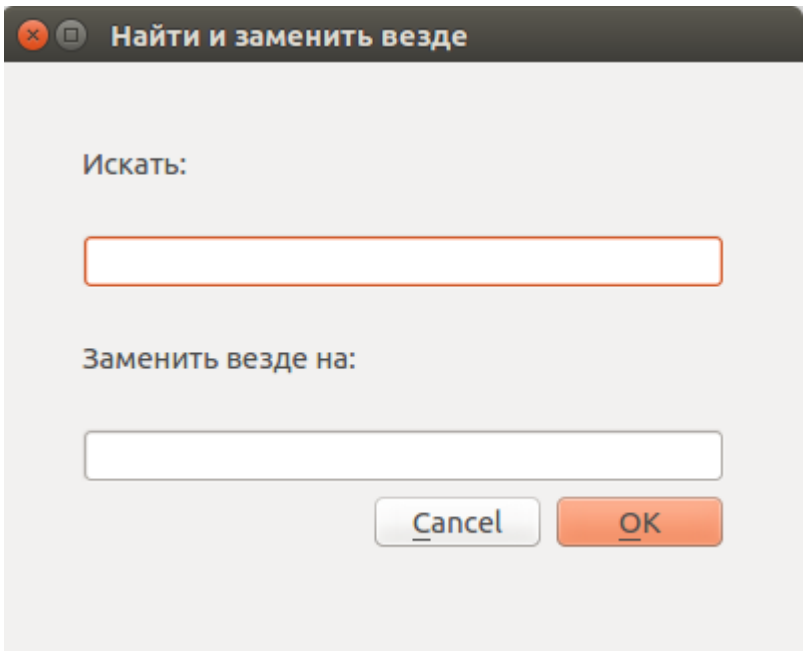

Рис. 21. Диалоговое окно “Найти и заменить везде”, русский вариант в Linux

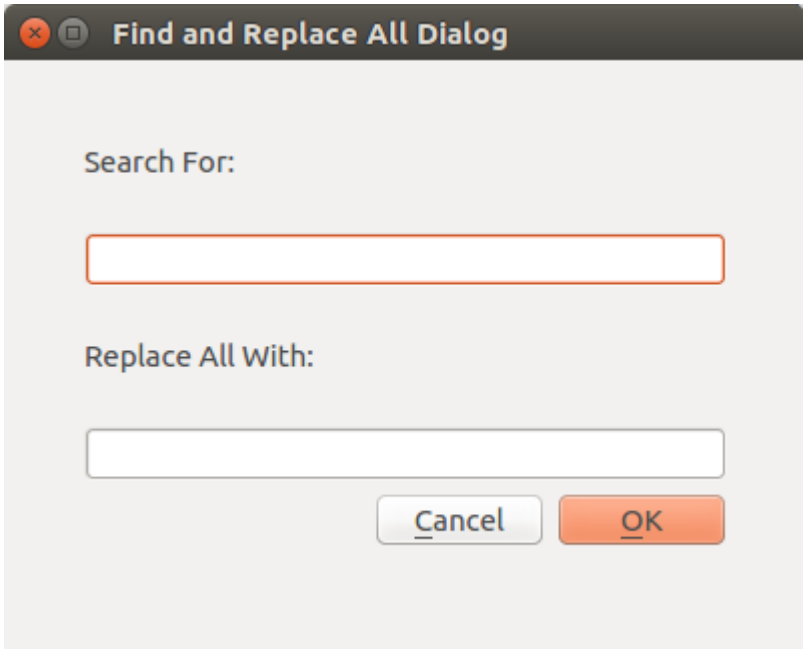

Рис. 22. Диалоговое окно “Найти и заменить везде”, английский вариант в Linux

5.2.7.2. Класс DialogReplaceAll

Он включает в себя меньшее количество слотов, а именно, слоты on_lineEdit_Find_editingFinished(), on_buttonBox_accepted(), on_buttonBox_rejected() и on_lineEdit_Replace_editingFinished().

5.2.7.3. Слот опции “Найти и заменить везде”

```
void MainWindow::on_actionFind_and_replace_all_triggered()
{
   textEdit_focused("Report");
   QString str = ui->textEdit_Report->toPlainText();
   if (str.isEmpty() || str.isNull()) {
      QMessageBox* warning = new QMessageBox(QMessageBox::Warning,
tr("Warning"), tr("The report is empty"));
      warning->exec();
      delete warning;
      return;
   }
   DialogReplaceAll dialogreplaceall;
   dialogreplaceall.exec();
   dialogreplaceall.show();

   if (!(dialogreplaceall.search.isEmpty() || dialogreplaceall.search.isNull())) {
      QString text = ui->textEdit_Report->toPlainText();
      text.replace(dialogreplaceall.search, dialogreplaceall.replace);
      ui->textEdit_Report->setText(text);
      QMessageBox::information(this, tr("Found and replaced"), tr("Find and
replace all are done"));
   }
}
```

5.3. Слоты для обработки сигналов подменю “Вид”

5.3.1. Слоты опций “Увеличить” и “Уменьшить”

Эти опции предназначены для увеличения или уменьшения шрифта в соответствующем текстовом поле или сразу во всех.

Слот для опции “Увеличить шрифт во всех окнах” имеет следующий код:

```
void MainWindow::on_actionZoom_in_all_windows_triggered()
{
   ui->textEdit_Questions->zoomIn();
   ui->textEdit_Report->zoomIn();
   ui->textEdit_TargetText->zoomIn();
   ui->textEdit_BaseText->zoomIn();

   ZoomQuestions++;
   ZoomReport++;
   ZoomTargetText++;
   ZoomBaseText++;
}
```

В слоте для опции “Уменьшить шрифт во всех окнах” zoomIn() везде заменяется на zoomOut(), а инкремент на декремент.

В слотах для опций увеличения или уменьшения шрифта в одном из текстовых полей вначале добавляется вызов метода textEdit_focused() с передачей строки со значением соответствующего поля. Кроме того, в них имеется вызов метода zoomIn() или zoomOut() только для соответствующего поля и инкремент или декремент только для соответствующей переменной.

5.3.2. Слоты опций “Шрифт”

Эти опции предназначены для выбора шрифта документа, открытого в соответствующем текстовом поле. Причем, за исключением отчета, выбор шрифта не сохраняется в соответствующем файле, но отображается на экране.

Ниже приводится код слота для опции “Шрифт вопросов”.

```
void MainWindow::on_actionFont_of_questions_triggered()
{
```

```
    textEdit_focused("Questions");
    bool ok;
    QFont font = QFontDialog::getFont(&ok, this); //Вызывается диалоговое
окно выбора шрифта
    if (ok) {
        ui->textEdit_Questions->setFont(font);
    } else return;
}
```

Слоты для опций выбора шрифта в других текстовых полях отличаются строкой, передаваемой в метод textEdit_focused(), и текстовым полем, в котором выбирается шрифт.

5.3.3. Слоты опций “Цвет шрифта” и “Цвет фона текста”

Эти опции предназначены для выбора цвета шрифта и цвета фона документа, открытого в соответствующем текстовом поле. За исключением отчета, выбор цвета шрифта не сохраняется в файле документа, но отображается на экране.

Ниже приводится код слота для опции “Цвет шрифта вопросов”

```
void MainWindow::on_actionText_color_of_questions_triggered()
{
    textEdit_focused("Questions");
    QColor color = QColorDialog::getColor(Qt::white, this, tr("Choose
Color")); //Вызывается диалоговое окно выбора цвета
    if(color.isValid()) {
        ui->textEdit_Questions->setTextColor(color);
    }
}
```

В опциях “Шрифт фона текста” метод setTextColor(color) заменяется на метод setTextBackgroundColor(color).

Кроме того, в зависимости от того, в каком из текстовых полей изменяется цвет шрифта или фона текста, изменяется значение строки, передаваемой в метод textEdit_focused() и текстовое поле.

5.3.4. Слоты опций “Цвет фона окна”

Эти опции позволяют выбрать и установить цвет фона в соответствующем текстовом поле или сразу во всех.

Код слота для опции “Цвет фона во всех окнах”:

```
void MainWindow::on_actionPalette_color_of_all_windows_triggered()
{
   QColor color = QColorDialog::getColor(Qt::white, this, tr("Choose
Color"));
   QPalette palette;
   palette.setColor(QPalette::Base, color);
   if(color.isValid()) {
      ui->textEdit_Questions->setPalette(palette);
      ui->textEdit_Report->setPalette(palette);
      ui->textEdit_TargetText->setPalette(palette);
      ui->textEdit_BaseText->setPalette(palette);
   }
}
```

Слоты для опций выбора цвета фона в одном из окон имеют дополнительно вызов метода textEdit_focused(), которому передается соответствующая строка. Кроме того, они содержат только одну строку с методом setPalette(palette) – для соответствующего текстового поля.

5.3.5. Слоты опций для выбора форматированного или неформатированного текста

5.3.5.1. Методы для форматированного и неформатированного текста в формате USFM или HTML

Несколько опций в подменю “Вид” предназначены для представления текста в одном из трех текстовых полей (кроме отчета) в форматированном или неформатированном виде.

Слоты для опций форматированного текста будут вызывать метод форматированного текста. Он позволяет представить текст файла в формате USFM (с которым работает программа Paratext) или в формате HTML в форматированном виде. При этом маркеры USFM или теги HTML будут скрыты.

Код метода приведен ниже.

```
void MainWindow::formatted(QString document_name) {
   QString file = "";

   if (document_name == "TargetText") { //Для текста перевода
      file = FilenameTargetText;
   }
   else if (document_name == "BaseText") { //Для русского текста
      file = FilenameBaseText;
   }
   else if (document_name == "Questions") { //Для файла с вопросами
      file = FilenameQuestions;
   }
   else return;

   if (file.isEmpty() || file.isNull()) {
      file = QFileDialog::getOpenFileName(this, tr("Open the file"), "",
                                 "Text files (*.txt);;Paratext files (*.sfm *.SFM);;\
HTML files (*.html *.htm)"); //Если файл не выбран, то будет выведено
диалоговое окно для выбора файла
   }

   if(!file.isEmpty())
   {
      QFile FileText(file);
      if(FileText.open(QFile::ReadOnly | QFile::Text)) //Метод повторно
открывает файл, поскольку он был закрыт сразу же после открытия
методом, открывшим файл
      {
         QTextStream in(&FileText);
         in.setCodec("UTF-8");
         QString text = in.readAll();
         FileText.close();
```

```
        QString text_converted = converter_to_html(text); //Вызов
конвертера

        if (document_name == "TargetText") { //Для текста перевода
            ui->textEdit_TargetText->setHtml(text_converted);
            FilenameTargetText = file;
        }
        else if (document_name == "BaseText") { //Для русского текста
            ui->textEdit_BaseText->setHtml(text_converted);
            FilenameBaseText = file;
        }
        else if (document_name == "Questions") { //Для файла с вопросами
            ui->textEdit_Questions->setHtml(text_converted);
            FilenameQuestions = file;
        }
    }
  }
}
```

Метод неформатированного текста позволяет представить текст файла в формате USFM (с которым работает программа Paratext) или в формате HTML в неформатированном виде. При этом маркеры USFM или теги HTML будут показаны.

Код этого метода отличается от кода метода форматированного текста тем, что нет вызова конвертера, а вместо метода setHtml() вызывается метод setPlainText().

5.3.5.2. Слоты опций “Форматированный (USFM / Paratext)” и “Неформатированный (USFM / Paratext)”

Эти опции позволяют выбирать представление текста на языке перевода, русского текста или файла с вопросами в форматированном или неформатированном виде. Код слота опции “Форматированный (USFM / Paratext)” для текста перевода:

```
void MainWindow::on_actionFormatted_target_text_triggered()
```

```
{
   textEdit_focused("TargetText");
   formatted("TargetText");
}
```

Опции для русского текста и файла с вопросами отличаются только строкой, передаваемой методам. Коды слотов для опции “Неформатированный (USFM / Paratext)” отличаются тем, что вместо метода formatted() вызывается метод unformatted().

5.3.5.3. Слот опции “Форматированный (Transcelerator)”

Эта опция есть только для файла с вопросами. Дело в том, что для создания списка вопросов есть дополнительный плагин к Paratext – Transcelerator, который имеет собственные маркеры. Если список вопросов обрабатывается согласно правилам для Paratext, хотя он создан в Transcelerator, то он будет обработан некорректно. Для этого создан специальный конвертер.

В слоте для данной опции происходит вызов этого конвертера, а также нет необходимости в разделении кода на метод и слот для обработки опции. Код этого слота:.

```
void MainWindow::on_actionFormatted_Transcelerator_triggered()
{
   textEdit_focused("Questions");

   QString file = FilenameQuestions;

   if (FilenameQuestions.isEmpty() || FilenameQuestions.isNull()) {
      file = QFileDialog::getOpenFileName(this, tr("Open the file"), "",
                                 "Text files (*.txt);;Paratext files (*.sfm *.SFM);;\
HTML files (*.html *.htm)");
   }

   if(!FilenameQuestions.isEmpty())
   {
```

```
        QFile FileText(file);
        if(FileText.open(QFile::ReadOnly | QFile::Text))
        {
            QTextStream in(&FileText);
            in.setCodec("UTF-8");
            QString text = in.readAll();
            FileText.close();
            QString text_converted = converter_from_transcelerator(text);

            ui->textEdit_Questions->setHtml(text_converted);
            FilenameQuestions = file;
        }
    }
}
void MainWindow::on_actionFormatted_Transcelerator_triggered()
{
    textEdit_focused("Questions");

    QString file = FilenameQuestions;

    if (FilenameQuestions.isEmpty() || FilenameQuestions.isNull()) {
        file = QFileDialog::getOpenFileName(this, tr("Open the file"), "",
                                    "Text files (*.txt);;Paratext files (*.sfm *.SFM);;\
HTML files (*.html *.htm)");
    }

    if(!FilenameQuestions.isEmpty())
    {
        QFile FileText(file);
        if(FileText.open(QFile::ReadOnly | QFile::Text))
        {
            QTextStream in(&FileText);
            in.setCodec("UTF-8");
            QString text = in.readAll();
            FileText.close();
            QString text_converted = converter_from_transcelerator(text);

            ui->textEdit_Questions->setHtml(text_converted);
            FilenameQuestions = file;
        }
    }
}
```

5.3.6. Слот опций “Кодировка”

Эта группа опций позволяет изменять кодировку документа в соответствующем текстовом поле. Это важно в том случае, если кодировка документа отличается от Unicode или UTF-8. В диалоговом окне можно выбрать одну из русских кодировок или одну из кодировок, используемых в языках Европы и Ближнего Востока.

5.3.6.1. Метод для изменения кодировки

Этот метод используется всеми четырьми слотами опций изменения кодировки. Поскольку он включает много кодировок, из которых можно выбрать, ниже приводится часть кода (не включающая все возможные кодировки).

```
void MainWindow::change_encoding(QString document_name) {
   QStringList items;
   //Создание и вывод диалогового окна для выбора кодировки
   items << "UTF-8" << "Unicode (UTF-16)" << tr("Windows-1251
(Cyrillic)") << tr("KOI8-R (Cyrillic)") <<
         tr("CP866 / IBM866 (Cyrillic)") << tr("ISO 8859-5 (Cyrillic)") <<
……

   bool ok;
   QString item = QInputDialog::getItem(this, tr("Changing encoding"),
tr("Choose encoding:"), items, 0, false, &ok);
   if (ok && !item.isEmpty()) {
      QString EncodingDocument; //Применение выбранной кодировки

      if (item == "UTF-8") EncodingDocument = "UTF-8";
      else if (item == "Unicode (UTF-16)") EncodingDocument = "Unicode";
      else if (item == tr("Windows-1251 (Cyrillic)")) EncodingDocument =
"Windows-1251";
      else if (item == tr("KOI8-R (Cyrillic)")) EncodingDocument = "KOI8-
R";
      else if (item == tr("CP866 / IBM866 (Cyrillic)")) EncodingDocument =
"IBM866";
```

```
        else if (item == tr("ISO 8859-5 (Cyrillic)")) EncodingDocument = "ISO 8859-5";
…….
        else return;

        if (document_name == "Report") EncodingReport = EncodingDocument; //В зависимости от того, кодировку какого документа надо изменить, устанавливается соответствующая кодировка для него
        else if (document_name == "Questions") EncodingQuestions = EncodingDocument;
        else if (document_name == "TargetText") EncodingTargetText = EncodingDocument;
        else if (document_name == "BaseText") EncodingBaseText = EncodingDocument;
        else if (document_name == "HTML") EncodingHTML = EncodingDocument; //Здесь обрабатывается случай изменения кодировки при отображении HTML кода, полученного в результате работы конвертера из USFM.

        else return;
    }
}
```

5.3.6.2. Слоты опций выбора кодировки

Код слота для опции “Кодировка текста вопросов”:

```
void MainWindow::on_actionEncoding_of_questions_triggered()
{
    textEdit_focused("Questions");
    change_encoding("Questions");
    open_questions(FilenameQuestions);
}
```

Слоты для опций кодировки других текстовых полей отличаютсястрокой, передаваемой в методы textEdit_focused() и change_encoding(), а также последней строкой, то есть, названием метода, вызываемого для открытия файла в соответствующем текстовом поле, и именем файла, передаваемым ему.

5.4. Слоты для обработки сигналов подменю “Вставка”

Эти слоты имеют достаточно сходный между собой код. Код слота для опции “Вставить название книги”:

```
void MainWindow::on_actionInsert_book_title_triggered()
{
   textEdit_focused("Report"); //Выделение текстового поля для отчета
   QString book_title = QInputDialog::getText(this, tr("Book title"), tr("Enter book title:"), QlineEdit::Normal); //Вызов диалога для ввода названия книги
   ui->textEdit_Report->insertPlainText(book_title + "\n"); //Вставка названия книги в отчет
}
```

Слоты для опций “Вставить номер” вместо метода getText() вызывают метод getInt(). Это слоты для опций “Вставить номер главы”, “Вставить номер стиха” и “Вставить номер вопроса”. Для опций “Вставить номера глав” и “Вставить номера стихов” диалоговое окно вызывается дважды – для ввода первой главы или первого стиха из диапазона и для ввода последней главы или последнего стиха из диапазонв.

Опции “Вставить время” и “Вставить заголовок” не требуют ничего вводить в диалоговое окно. В группу “Вставить время” входят две опции – “Вставить время начала работы” и “Вставить время окончания работы”. Код слота опции “Вставить время начала работы”:

```
void MainWindow::on_actionInsert_time_of_work_start_triggered()
{
   textEdit_focused("Report");
   QDateTime work_start = QDateTime::currentDateTime(); //Получение текущего времени, которое в данном случае будет временем начала работы
   ui->textEdit_Report->insertPlainText(tr("Work began at:") + " " + work_start.toString("dd/MM/yyyy hh:mm:ss") + "\n"); //Вставка времени в отчет в определенном формате
```

```
}
```

Код слота опции “Вставить время окончания работы” отличается тем, что объект QDateTime – work_end, а также вставляемой в отчет строкой – tr("Work ended at:").

Код слота опции “Вставить заголовок в отчет”:

```
void MainWindow::on_actionInsert_report_title_triggered()
{
   textEdit_focused("Report");
   ui->textEdit_Report->insertHtml(tr("<b>The comprehension testing report</b><br>") + "\n"); //Добавляется заголовок жирным шрифтом “Отчет по апробации”
}
```

5.5. Слоты для обработки сигналов подменю “Ответ респондента” и подменю “Комментарий”

Эти слоты вставляют выбранную строку в текст отчета. Для примера приведен код слота для опции “Респондент дал правильный ответ”:

```
void MainWindow::on_actionRespondent_answered_correctly_triggered()
{
   textEdit_focused("Report");
   ui->textEdit_Report->insertPlainText(tr("Respondent answered correctly") + ".\n");
}
```

Остальные слоты работают аналогично, вставляя выбранную строку в отчет.

5.6. Слоты для обработки сигналов подменю “Опции”

5.6.1. Слот опций “Язык интерфейса”

Сюда относятся слоты для опций “Английский” и “Русский”. Код слота для опции “Английский”:

```
void MainWindow::on_actionEnglish_triggered()
{
   language = "English";
   QMessageBox::information(this, tr("Language change"),
                            tr("Interface language will be changed after the program
restarts.")); //Вызов сообщения о том, что язык интерфейса будет изменен
после перезапуска программы
}
```

Код слота для опции “Русский” отличается только тем, что строка "English" заменяется на "Russian".

5.6.2. Слоты опций автоматической вставки времени, заголовка и загрузки файлов

Опция “Автоматическая вставка времени” позволяет установить автоматическую вставку времени начала в отчет при каждом открытии файла отчета и времени окончания работы при каждом закрытии файла отчета или закрытии программы. Она также позволяет отменить ранее установленную автоматическую вставку времени в отчет.

```
void MainWindow::on_actionInsert_time_automatically_triggered()
{
   textEdit_focused("Report");
   QMessageBox* messagebox = new QMessageBox
(QmessageBox::Question, tr("Insert time"), tr("Insert time of beginning and
end of work automatically?"), QMessageBox::Yes | QMessageBox::No,
this); //Вызов диалога для выбора между автоматической вставкой
времени и ее отменой
   messagebox->setButtonText(QMessageBox::Yes, tr("Yes"));
   messagebox->setButtonText(QMessageBox::No, tr("No"));
   int n = messagebox->exec();
   delete messagebox;

   if (n == QMessageBox::Yes) {
      TimeAuto = true;
      QMessageBox::information(this, tr("Insert time"),
tr("Time of beginning and end of work will be inserted automatically."));
//Вызов сообщения о том, что время будет вставляться автоматически
```

```
    } else {
      TimeAuto = false;
      QMessageBox::information(this, tr("Insert time"),
tr("Time of beginning and end of work will not be inserted
automatically.")); //Вызов сообщения о том, что время не будет
вставляться автоматически
    }
}
```

Опция “Автоматическая вставка заголовка” позволяет автоматически вставлять в новый отчет заголовок “Отчет по апробации”. Она также позволяет отменить автоматическую вставку заголовка.

Код слота этой опции отличается от предыдущей только текстом вызываемого сообщения, а также переменной TitleAuto вместо TimeAuto, которая сохраняется в файле настроек.

По умолчанию, при каждом повторном запуске программы в соответствующие текстовые поля будут загружаться тексты открытых ранее файлов – вопросов по апробации, отчета по апробации, текста перевода и текста на русском языке. Для этого названия файлов и пути к ним сохраняются в файле настроек программы. Опция “Автоматическая загрузка файлов” позволяет отменить автоматическую загрузку этих файлов или восстановить ее после отмены.

Код слота этой опции также отличается от кода слота опции “Автоматическая вставка времени” только текстом сообщения и переменной, сохраняемой в файле настроек – FilesNotLoadAuto вместо TimeAuto.

5.6.3. Слоты опций “USFM -> HTML”

Эти опции позволяют просматривать HTML код, который производит конвертер USFM в HTML. Опция для просмотра HTML

кода, создаваемого другим конвертером – конвертером из Transcelerator в HTML код нет, поскольку в такой опции нет практической необходимости. Просмотр HTML кода, производимого конвертером USFM в HTML, может быть полезен для создания HTML файлов из файлов Paratext и размещения их на сайте, но Transcelerator сам может генерировать HTML файлы, поэтому нет необходимости в опции просмотра HTML кода конвертера.

5.6.3.1. Метод “USFM -> HTML”

Код этого метода:

```
void MainWindow::USFM_to_HTML(QString document_name) {
   QString file = "";

   if (document_name == "TargetText") {
      file = FilenameTargetText;
   }
   else if (document_name == "BaseText") {
      file = FilenameBaseText;
   }
   else if (document_name == "Questions") {
      file = FilenameQuestions;
   }
   else return;

   if (file.isEmpty() || file.isNull()) {
      file = QFileDialog::getOpenFileName(this, tr("Open the file"), "",
                                 "Paratext files (*.sfm *.SFM)");
   }

   if(!file.isEmpty() && (file.endsWith(".sfm") || (file.endsWith(".SFM"))))
   {
      QFile FileText(file);
      if(FileText.open(QFile::ReadOnly | QFile::Text))
      {
         QTextStream in(&FileText);
         if (EncodingHTML == "UTF-8") in.setCodec("UTF-8");
```

```
        else if (EncodingHTML == "Unicode") in.setCodec("Unicode");
        else if (EncodingHTML == "Windows-1251") in.setCodec("Windows-
1251");
        else if (EncodingHTML == "KOI8-R") in.setCodec("KOI8-R");
        else if (EncodingHTML == "IBM866") in.setCodec("IBM866");
        else if (EncodingHTML == "ISO 8859-5") in.setCodec("ISO 8859-
5");
    ……
        else in.setCodec("UTF-8");

        QString text = in.readAll();
        FileText.close();

        QString text_converted = converter_to_html(text);

        if (document_name == "TargetText") {
           ui->textEdit_TargetText->setPlainText(text_converted);
           FilenameTargetText = file;
        }
        else if (document_name == "BaseText") {
           ui->textEdit_BaseText->setPlainText(text_converted);
           FilenameBaseText = file;
        }
        else if (document_name == "Questions") {
           ui->textEdit_Questions->setPlainText(text_converted);
           FilenameQuestions = file;
        }
      }
   }
}
```

5.6.3.2. Слоты опций “USFM -> HTML”

Код слота для опции “Файл вопросов USFM -> HTML”:

```
void MainWindow::on_actionQuestionsUSFM_HTML_triggered()
{
   textEdit_focused("Questions");
   USFM_to_HTML("Questions");
}
```

Слоты для обработки опций для других текстовых полей имеют аналогичный код и отличаются только текстовой строкой, передаваемой в метод textEdit_focused() и в функцию USFM_to_HTML().

5.6.3.3. Слот опции “Кодировка USFM -> HTML”

Эта опция позволяет изменять кодировку при просмотре HTML кода. Для простоты сделана одна опция, изменяющая настройку кодировки при просмотре HTML сразу в трех текстовых полях.

```
void MainWindow::on_actionEncoding_for_USFM_HTML_triggered()
{
   change_encoding("HTML");
}
```

5.7. Слот для обработки сигнала подменю “Справка”

Это подменю содержит только одну опцию - “О программе”, которая выводит краткое сообщение о программе и ее назначении. В дальнейшем возможно добавление справочной системы к этому подменю. Код слота опции “О программе”:

```
void MainWindow::on_actionAbout_triggered()
{
   QString about_text;
   about_text  = tr("Program for comprehension testing of Bible texts") + "\n";
   about_text += tr("Version 1.0") + "\n";
   about_text += tr("Developer: Aleksandr Migunov, 2021");
   QMessageBox::about(this, tr("About Program"), about_text);
}
```

Глава 6. Разработка программного кода для обработки опций панели инструментов

Большинство опций панели инструментов дублируют соответсвующие опции главного меню, поэтому новые слоты для них не нужны. Нужны только слоты для опций, которых нет в главном меню. Все эти опции используются для форматирования текста отчета.

6.1. Слоты опций “Жирный шрифт”, “Курсив” и “Подчеркивание”

Программный код слотов для обработки этих опций достаточно похож. Для примера ниже приведен код слота опции “Жирный шрифт”.

```
void MainWindow::on_actionTextBold_triggered()
{
   textEdit_focused("Report");

   if (TextBold == false) { //При каждом нечетном нажатии на соответствующую кнопку устанавливается жирный шрифт
      ui->textEdit_Report->setFontWeight(QFont::Bold);
      TextBold = true;
   }
   else { //При каждом четном нажатии на соответствующую кнопку убирается жирный шрифт
      ui->textEdit_Report->setFontWeight(QFont::Normal);
      TextBold = false;
   }
}
```

Код слотов опций “Курсив” и “Подчеркивание” отличается, во-первых, тем, что вместо переменной TextBold используются соответственно переменные TextItalic и TextUnderline.

Во-вторых, setFontWeight(QFont::Bold) заменяется для опции “Курсив” на setFontItalic(true), а для опции “Подчеркивание – на setFontUnderline(true). А setFontWeight(QFont::Normal) заменяется соответственно на setFontItalic(false) и на setFontUnderline(false).

6.2. Слоты опций “Выравнивание”

Слоты для этих опций являются однотипными. Для примера ниже приведен код слота для опции “Выравнивание слева”.

```
void MainWindow::on_actionAlignLeft_triggered()
{
   textEdit_focused("Report");
   ui->textEdit_Report->setAlignment(Qt::AlignLeft);
}
```

Соответственно, слоты опций для других видов выравнивания отличаются тем, что код слота для выравнивания справа в последней строке имеет Qt::AlignRight, по центру – Qt::AlignCenter, с обеих сторон – Qt::AlignJustify.

Глава 7. Разработка программного кода для контекстных меню

Для создания контекстного меню для каждой опции создается действие – объект QAction, который затем соединяется со слотом для его обработки (все слоты – это те же слоты, что и для обработки опций главного меню). После этого все действия добавляются к контекстному меню – объекту QMenu.

Поскольку контекстные меню имеют сходный код (отличаясь конкретными опциями и слотами), то приводить весь код нет необходимости. Поэтому ниже приведен фрагмент кода контекстного меню для текстового поля вопросов.

```
void MainWindow::on_textEdit_Questions_customContextMenuRequested()
{
   textEdit_focused("Questions");

   QAction *actionOpen = new QAction(tr("Open"), this);
   connect(actionOpen, SIGNAL(triggered()), this,
SLOT(on_actionOpen_questions_triggered()));

   QAction *actionClose = new QAction(tr("Close"), this);
   connect(actionClose, SIGNAL(triggered()), this,
SLOT(on_actionClose_questions_triggered()));

…….

   QMenu *contextMenu = new QMenu(this);
   contextMenu->addAction(actionOpen);
   contextMenu->addAction(actionClose);
   contextMenu->insertSeparator(0);
……...
   contextMenu->exec(QCursor::pos());
}
```

Глава 8. Разработка конвертеров

Поскольку C++, в отличие от C#, поддерживает не только объектно-ориентированное, но и процедурное программирование, для создания двух конвертеров не был создан новый класс, поскольку в этом не было особой необходимости. Таким образом, конвертеры – это функции, а не методы класса.

8.1. Конвертер из USFM/Paratext в HTML

Файлы в формате USFM имеют специальные маркеры форматирования, которые используются программой Paratext для визуального представления. Однако Paratext не является программой с открытым кодом. Хотя файлы USFM (с расширением .sfm) можно открыть любым текстовым редактором, такое представление является не очень удобным для апробаторов, поскольку показывает все маркеры USFM и неотформатированный текст.

Для того чтобы эти файлы можно было просматривать в форматированном виде, в данной программе был разработан специальный конвертер из USFM в HTML. Qt хорошо умеет работать с HTML и представляет этот формат как форматированный текст.

Однако стоит заметить, что маркеры USFM достаточно сильно отличаются от тегов HTML и однозначного соответствия между ними нет. Прежде всего маркеры USFM предназначены в первую очередь для разметки книг, которые готовятся к публикации или уже опубликованы в печатном виде. Существует большое количество различных маркеров USFM. Их количество постоянно меняется – всё время добавляются новые, а некоторые старые отменяются. При этом многие маркеры USFM используются редко или являются факультативными.

В дополнение к маркерам USFM существуют таблицы стилей для настройки отображения каждого маркера на экране. Эти таблицы стилей можно настраивать, но они предоставляют намного меньшие возможности для настройки, чем, например, CSS.

Однако для создания конвертера из USFM в HTML для использования в разрабатываемой программе нет необходимости учитывать все нюансы маркеров USFM. Достаточно того, чтобы получаемый HTML код отображал форматирование текста, заданное с помощью маркеров USFM приблизительно так же, как исходный текст отображается в программе Paratext. Кроме того, нет практической необходимости включать в конвертер редко используемые маркеры.

Таким образом, в данной программе был разработан упрощенный конвертер, позволяющий корректно обрабатывать основные маркеры USFM и осуществлять форматирование, которое удовлетворительно для практических задач.

Ниже приводится код этого конвертера в сокращении, поскольку он содержит много однотипных строк для обработки разных маркеров.

```
QString converter_to_html (QString str) {
   str.replace("<", "##lt;"); //Поскольку знаки “больше” и “меньше” используются в тегах HTML, их надо заменить на соответствующие кодовые последовательности. Однако амперсанды будут использоваться в конвертере, поэтому сначала вместо амперсандов ставятся двойные хеши, которые в конце программы будут заменены на одинарные амперсанды.
   str.replace(">", "##gt;");
   str.replace("\\", "&&"); //Поскольку все маркеры USFM начинаются с обратного слеша, который в языках программирования служит для обозначения эскейп-последовательностей, то для устранения проблем при обработке маркеров USFM, прежде всего все обратные слеши заменяются на двойные амперсанды (двойные – чтобы не возникло проблем, если в тексте встретятся обычные амперсанды).
```

```
    str.replace("&&ide", "<br>"); //Порядок маркеров имеет большое значение – маркер \ide должен стоять перед \id, чтобы он был корректно обработан. Если поменять порядок, то от маркера \ide останется буква “e”, и он не будет правильно обработан.
    str.replace("&&id", "<p>");
    str.replace("&&sts", "<br>");
    str.replace("&&ili1", "<li>");
    str.replace("&&ili", "<li>");
    str.replace("&&pi1", "<p>");
    str.replace("&&pi", "<p>");
……
    //добавление закрывающих тегов, поскольку ряд соответствующих закрывающих маркеров не используется (и не требуется) в USFM
    QRegExp rgx1("<h1> ([^&<]+)");
    str.replace(rgx1, "<h1>\\1</h1>");
    QRegExp rgx2("<h2> ([^&<]+)");
    str.replace(rgx2, "<h2>\\1</h2>");
……
    str.replace("##lt;", "&lt;");  //Повторная корректировка
    str.replace("##gt;", "&gt;");
    return str;
}
```

8.2. Конвертер из Transcelerator в HTML

Этот конвертер намного короче предыдущего, поскольку для него предусмотрено намного меньше маркеров. Хотя файлы, создаваемый Transcelerator, имеют то же расширение, что и Paratext – .sfm, но эти маркеры другие и не входят в стандартный USFM.

```
QString converter_from_transcelerator(QString str) {
    str.replace("\n", "%%"); //Корректировка для завершающих HTML тегов
    str.replace("\\_sh", "<p>sh");
    str.replace("\\rf", "<h1>");
    QRegExp rgx20("<h1> ([^%]+)");
    str.replace(rgx20, "<h1>\\1</h1>");
……...
    str.replace("%%", "\n"); //Повторная корректировка для пробелов
    return str;
}
```

Глава 9. Компиляция, сборка и тестирование программы

Программа была скомпилирована и собрана для Linux и для Windows и протестирована в Ubuntu 16.04, 18.04, 20.04 и в Windows Vista, Windows 7, Windows 10.

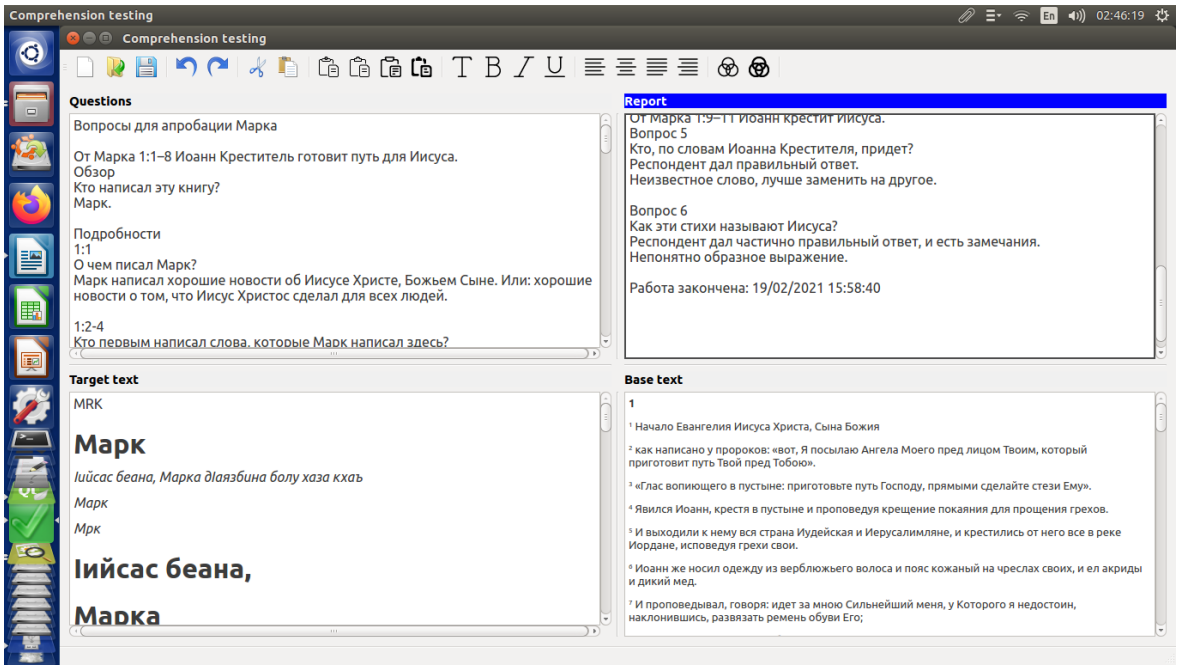

Рис. 23. Работа программы в Linux (с английским интерфейсом)

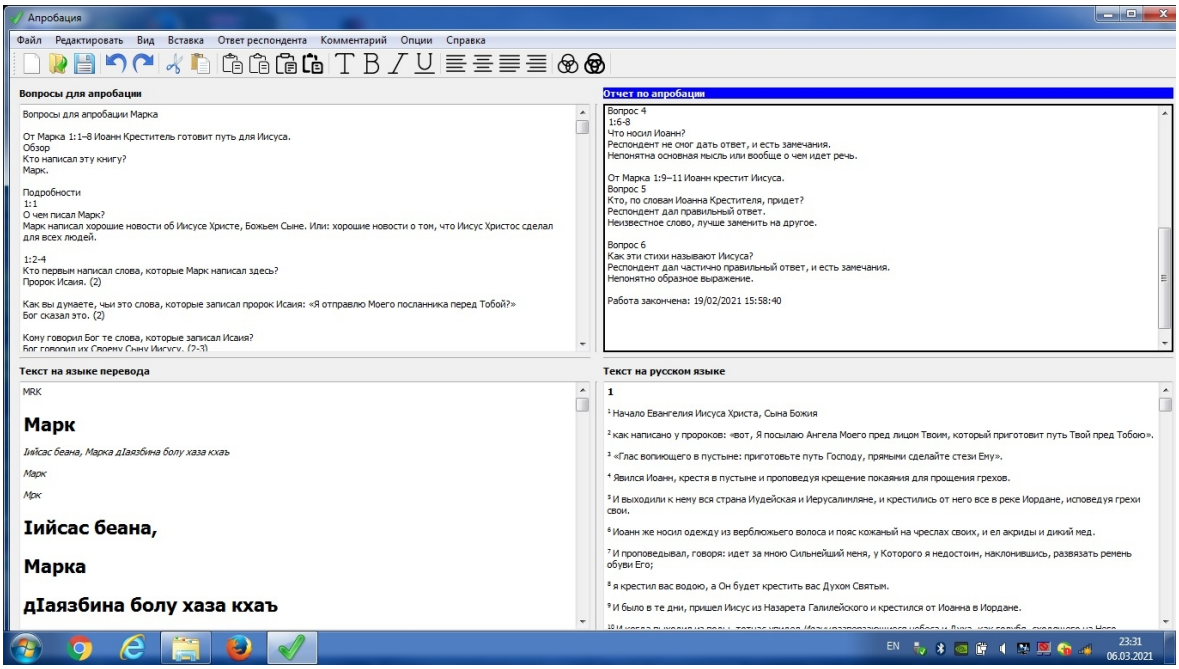

Рис. 24. Работа программы в Windows (с русским интерфейсом)

Кроме того, версия для Windows была также установлена (через Wine) и протестирована на Linux в режимах совместимости с Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1 и Windows 10.

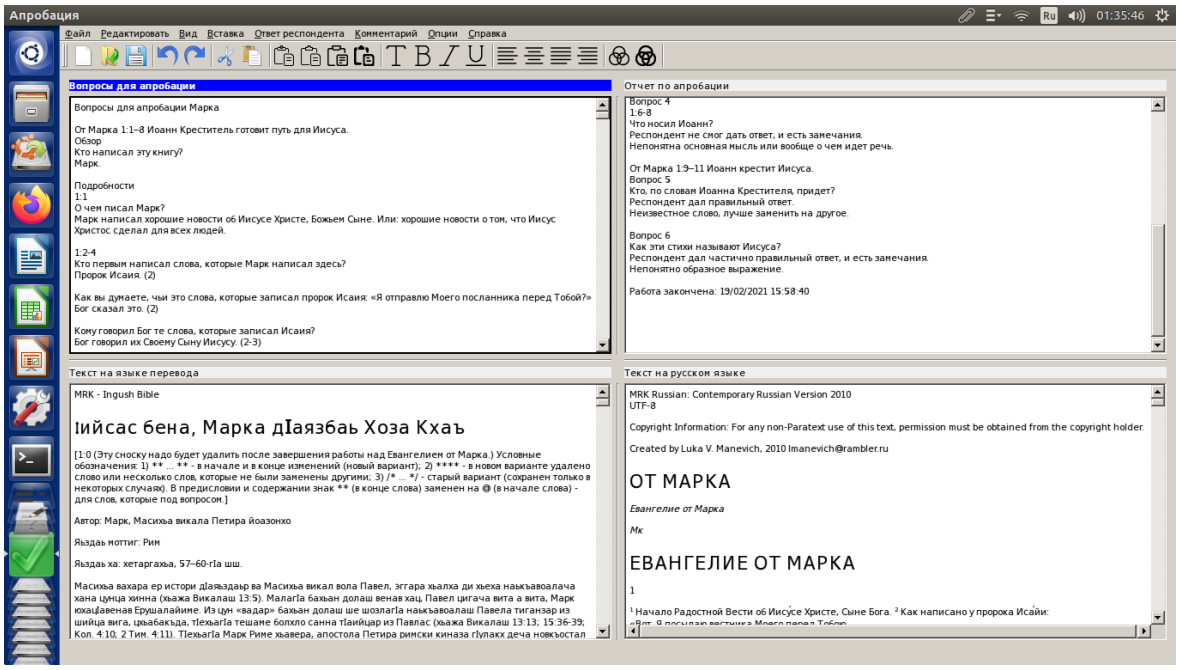

Рис. 25. Работа версии программы для Windows в Linux через Wine (эмулятор для Windows), русский интерфейс

Как видно из рис. 23, 24, в Linux и Windows есть внешние различия в интерфейсе программы (версия для Windows, запущенная в Linux через Wine, соответственно имеет интерфейс очень близкий к интерфейсу в Windows). Но за исключением этих внешних различий работа программы в обеих операционных системах полностью идентична. Причем исходный код программы для обеих систем абсолютно одинаков. Таким образом, разработанная программа является полностью кроссплатформенной.

Для Windows были созданы 64-битная и 32-битная версии. Сначала программа была скомпилирована в Qt Creator (в 32-битной и 64-

битной версиях Windows). Далее, все необходимые DLL были собраны вручную в папки с полученными исполняемыми файлами программы. После этого были созданы инсталляторы с помощью Inno Setup. Для Linux процедура была несколько другая. После компиляции программы в Qt Creator, с помощью linuxdeployqt был получен файл AppImage, включающий все необходимые библиотеки и не требующий установки. Кроме того, с помощью Qt Installer Framework был создан инсталлятор.

Тестирование программы в Linux и Windows прошло успешно. Все обнаруженные ошибки были исправлены. Поскольку в процессе разработки программы проводилось тестирование отдельных частей программы и исправлялись ошибки, на последнем этапе тестирования ошибок было найдено немного.

Заключение

Целью данной работы была разработка программы для апробации, которая бы была кроссплатформенной и работающей в Linux и Windows. Эта цель была достигнута. С использованием C++ Qt была создана кроссплатформенная программа, позволяющая упростить проведение апробации и значительно ускорить составление отчета по апробации, а также содержащая ряд других полезных опций. Эта программа была успешно протестирована.

В дальнейшем исходный код программы планируется использовать при разработке других программ, в частности, программы для богословского редактирования (которое, как и апробация, является одним из этапов работы над переводом Библии). Одной из главных задач при богословком редактировании является создание списка вопросов и замечаний для переводчика той или иной книги Библии. После небольшой модификации данной программы, а именно замены типовых ответов респондента и комментариев по апробации на типовые замечания богословского редактора, а также изменения назначения и, возможно, количества текстовых полей, программа может с успехом использоваться для создания списка вопросов и комментариев богословского редактирования.

Кроме того, конвертер из USFM в HTML может быть использован в дальнейшем для добавления новых книг на ингушский библейский сайт, разработанный ранее.

Таким образом, с одной стороны, разработанная программа является полезной сама по себе, поскольку она обеспечивает апробаторов удобным инструментарием для работы. А с другой стороны, разработанный программный код будет использован повторно в дальнейшем.

Список использованной литературы

1. Шлее М. Qt 5.10. Профессиональное программирование на C++. / М. Шлее. – СПб.: БХВ-Петербург, 2019. – 1072 с.: ил.

2. Бланшет Ж. Qt 4: программирование GUI на C++, 2-е изд., доп. / Ж. Бланшет, М. Саммерфилд. – Пер. с англ. – М.: КУДИЦ-ПРЕСС, 2008. – 736 с.

3. Саммерфилд М. Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++. / М. Самерфилд. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 560 с.: ил.

4. Земсков Ю.В. Qt 4 на примерах. / Ю.В. Земсков. – СПб.: БХВ-Петербург, 2008. – 608 с.: ил.

5. Боровский А.Н. Qt 4.7+. Практическое программирование на C++. / А.Н. Боровский. – СПБ.: БХВ-Петербург, 2012. – 496 с.: ил.

6. Павловская Т.А. C/C++. Процедурное и объектно-ориентированное программирование: Учебник для вузов. Стандарт 3-го поколения. / Т.А. Павловская. – СПб.: Питер, 2018. – 496 с.: ил.

7. Павловская Т.А. C/C++. Программирование на языке высокого уровня. / Т.А. Павловская. – СПб.: Питер, 2019. – 464 с.: ил.

8. Прохоренок Н.А. Python 3 и PyQt 5. Разработка приложений. – 2-е изд., перераб. и доп. / Н.А. Прохоренок, В.А. Дронов. – СПб.: БХВ-Петербург, 2019. – 832 с.: ил.

9. Мансуров К.Т. Основы программирования в среде Lazarus./ К.Т. Мансуров. – [Б. м.], 2010. – 772 с.: ил.

10. USFM (Unified Standard Format Markers) 2.4: User Reference. – United Bible Societies, 2013. – 97 с.

Оглавление

	№ страницы
Введение	2
Глава 1. Постановка задачи и определение требований	3
Глава 2. Выбор IDE и фреймворка для разработки	4
Глава 3. Разработка графического интерфейса программы	6
3.1. Центральная часть (центральный виджет) главного окна	6
3.2. Главное меню	8
3.3. Панель инструментов	16
3.4. Контекстные меню	17
3.5. Внешний вид программы (главного окна)	21
Глава 4. Разработка программного кода методов, запускаемых при открытии программы и при ее закрытии	23
4.1. Разработка главной функции программы	23
4.2. Конструктор главного окна	24
4.3. Методы для записи и чтения файла настроек	25
4.4. Метод для выделения одного из текстовых полей и метки с заголовком к нему	28
4.5. Метод для открытия файлов, с которыми пользователь работал в предыдущий раз	29
4.6. Методы для открытия файлов	30
4.7. Метод для установки масштаба шрифта в текстовых полях при запуске программы	33
4.8. Метод закрытия программы	34
4.9. Метод закрытия файла отчета	35
4.10. Слоты выбора текстовых полей	36
Глава 5. Разработка программного кода для обработки опций главного меню	37

5.1. Слоты для обработки сигналов подменю “Файл” 37
5.1.1. Слоты опций “Открыть” 37
5.1.2. Слоты опций “Закрыть” 38
5.1.3. Слот опции “Создать новый файл отчета” 39
5.1.4. Слот опции “Сохранить отчет” 39
5.1.5. Слот опции “Сохранить отчет как” 40
5.1.6. Слоты опций “Печать” 41
5.1.7. Слоты опций “Предварительный просмотр” 42
5.1.8. Опции “Конвертировать в PDF” и “Конвертировать в ODT” 42
5.1.8.1. Методы для конвертирования в PDF и в ODT 43
5.1.8.2. Слоты опций конвертирования в PDF / ODT” 43
5.1.9. Слот опции “Выйти из программы” 45
5.2. Слоты для обработки сигналов подменю “Редактировать” 46
5.2.1. Слоты опций “Отменить”, “Повторить”, “Вырезать” и “Вставить” 46
5.2.2. Слоты опций “Копировать” 46
5.2.3. Слоты опций “Копировать и вставить” 47
5.2.4. Слот опции “Удалить” 47
5.2.5. Слоты опций “Выделить всё” 48
5.2.5. Опции “Найти” 48
5.2.5.1. Диалоговое окно “Поиск” 48
5.2.5.2. Класс DialogFind 49
5.2.5.3. Метод для поиска 50
5.2.5.4. Слоты опций поиска 54
5.2.6. Опция “Найти и заменить” 54
5.2.6.1. Диалоговое окно “Поиск и замена” 54
5.2.6.2. Класс DialogReplace 56

5.2.6.3. Слот опции “Найти и заменить” 56
5.2.7. Опция “Найти и заменить везде” 56
5.2.7.1. Диалоговое окно “Найти и заменить везде” 57
5.2.7.2. Класс DialogReplaceAll 58
5.2.7.3. Слот опции “Найти и заменить везде” 58
5.3. Слоты для обработки сигналов подменю “Вид” 58
5.3.1. Слоты опций “Увеличить” и “Уменьшить” 58
5.3.2. Слоты опций “Шрифт” 59
5.3.3. Слоты опций “Цвет шрифта” и “Цвет фона текста” 60
5.3.4. Слоты опций “Цвет фона окна” 61
5.3.5. Слоты опций для выбора форматированного или неформатированного текста 61
5.3.5.1. Методы для форматированного и неформатированного текста в формате USFM или HTML 61
5.3.5.2. Слоты опций “Форматированный (USFM / Paratext)” и “Неформатированный (USFM / Paratext)” 63
5.3.5.3. Слот опции “Форматированный (Transcelerator)” 64
5.3.6. Слот опций “Кодировка” 66
5.3.6.1. Метод для изменения кодировки 66
5.3.6.2. Слоты опций выбора кодировки 67
5.4. Слоты для обработки сигналов подменю “Вставка” 68
5.5. Слоты для обработки сигналов подменю “Ответ респондента” и подменю “Комментарий” 69
5.6. Слоты для обработки сигналов подменю “Опции” 69
5.6.1. Слот опций “Язык интерфейса” 69
5.6.2. Слоты опций автоматической вставки времени, заголовка и загрузки файлов 70
5.6.3. Слоты опций “USFM -> HTML” 71

5.6.3.1. Метод “USFM -> HTML” 72
5.6.3.2. Слоты опций “USFM -> HTML” 73
5.6.3.3. Слот опции “Кодировка USFM -> HTML” 74
5.7. Слот для обработки сигнала подменю “Справка” 74
Глава 6. Разработка программного кода для обработки опций панели инструментов 75
6.1. Слоты опций “Жирный шрифт”, “Курсив” и “Подчеркивание” 75
6.2. Слоты опций “Выравнивание” 76
Глава 7. Разработка программного кода для контекстных меню 77
Глава 8. Разработка конвертеров 78
8.1. Конвертер из USFM/Paratext в HTML 78
8.2. Конвертер из Transcelerator в HTML 80
Глава 9. Компиляция, сборка и тестирование программы 81
Заключение 84
Список использованной литературы 85
Приложение 1. Листинги кода программы
Приложение 2. Проект программы в Qt Creator