Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____РК_____

КАФЕДРА ___Компьютерные системы автоматизации производства РК-9_____

_____

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к курсовому проекту на тему:

___Модернизация системы документации РДО_____

_____

_____

_____

_____

_____

_____

Студент _____ _____ _Клеванец И.С._

(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____ _Урусов А.В.___

(Подпись, дата) (И.О.Фамилия)

Москва, 2012

Государственное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой __РК9____
(Индекс)
_____ _____
(И.О.Фамилия)
« ___ » _____ 20 __ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине_____

Модернизация системы документации РДО
(Тема курсового проекта)

Студент__*Клеванец И.С. РК9-101*_____
(Фамилия, инициалы, индекс группы)

График выполнения проекта: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

1. Техническое задание

2. ***Оформление курсового проекта***
2.1. Расчетно-пояснительная записка на ___ листах формата А4.
2.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.)_____
лист 1— постановка задачи;
лист 2 – диаграмма компонентов;
лист 3— диаграммы классов, синтаксическая диаграмма;
лист 4— алгоритм работы генератора;
лист 5—результаты;
Дата выдачи задания « » _____ 2012г.

Руководитель курсового проекта _____ *Урусов А.В.*
(Подпись, дата) (И.О.Фамилия)

Студент _____ *Клеванец И.С.*
(Подпись, дата) (И.О.Фамилия)

Примечание:
1. Задание оформляется в двух экземплярах; один выдаётся студенту, второй хранится на кафедре.

ОГЛАВЛЕНИЕ

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1. Введение

Любой программный продукт должен быть доступным для эксплуатации пользователем: с помощью интуитивно понятного интерфейса и исчерпывающей документации. Для любого языка моделирования или программирования понятие интерфейса – не более чем надстройка над самим языком, и потому язык РДО без документации (справки) не может быть изучен и позже использован разработчиком имитационных моделей.

1.2. Основания для разработки

Программный продукт РДО обладает телом документации, не адекватным текущему состоянию РДО. Файловая структура документации РДО обладает плоской структурой, что неверно: количество файлов слишком велико, чтобы хранить их плоским списком. Формат документации (CHM) неудобен для разработчиков РДО, которым необходимо вносить свои изменения в документацию. К тому же необходимость хранения файлов именно плоским списком – одно из условий работы документации в формате CHM. Наличие больших (и потому неудобных) разделов документации. Отсутствие сайта документации.

1.3. Назначение разработки

- привести тело документации в валидное состояние;
- перейти на формат документации, не имеющий недостатков формата CHM;
- разбить большие разделы документации на более лаконичные;
- создать сайт документации;
- создать ПО сопровождения сайта.

1.4. Требования к программе или программному изделию

а) Требования к функциональным характеристикам:

- вызов документации РДО, в том числе контекстно-чувствительный;
- полноценная работа с документацией в новом формате:
  - навигация по содержанию;
  - поиск по ключевым словам;
  - поиск по телу документации;
- полноценная работа со документацией через сайт:
  - навигация по содержанию;
  - поиск по ключевым словам;
- кросс-платформенная доступность документации;
- простое использование ПО сопровождения сайта.

б) Требования к надежности:

- стабильная работа РДО при вызове документации;
- стабильная работа системы документации;

- стабильная работа сайта;
- стабильная работа ПО сопровождения сайта.

в) Условия эксплуатации:
- соответствуют условиям эксплуатации РДО;
- последние стабильные версии распространенных браузеров.

г) Требования к составу и параметрам технических средств:
- полностью соответствуют требованиям к РДО;

д) Требования к информационной и программной совместимости:
- полностью соответствуют требованиям к РДО.

1.5. **Требования к программной документации**:
- инструкция для использования ПО сопровождения сайта.

1.6. **Стадии и этапы разработки**:
а) исправление ошибок и недоработок в рамках формата CHM;
б) выбор формата документации, соответствующего требованиям, заявленным в ТЗ;
в) переход на другой формат документации;
г) рефакторинг файловой структуры документации;
д) корректировка тела документации;
е) разработка концепции сайта;
ж) разработка ПО для сопровождения сайта;
з) организация автоматического обновления сайта;
и) разработка контекстно-чувствительного вызова документации.

1.7. **Порядок контроля и приемки**:
а) запуск РДО-студии и вызов документации;
б) создание сайта с помощью ПО сопровождения;
в) доступ к сайту документации через интернет.

2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

2.1. **Тело документации**

Язык имитационного моделирования РДО развивается бурными темпами, и поэтому не все изменения нашли место в документации. Встречаются архаизмы, недостает нововведений. Есть нововведения, которые затронули несколько разделов, но обновлены были не все.

К тому же существуют проблемы исключительно технического плана: проблемы с тэгами HTML, скриптами и банальными неработающими ссылками.

2.2. **Система документации**

2.2.1. Описание существующей системы

Система документации РДО основана на формате CHM (Microsoft Compressed/ Compiled HTML Help). Это проприетарный формат файлов контекстной документации, разработанный корпорацией Microsoft. Представляет собой особым образом скомпилированный файл-архив, содержащий HTML-страницы с дополнительной информацией.

Для получения итогового файла документации необходимо иметь проект документации. Сам текст документации хранится в HTML-формате, данные для поиска по телу документации: по ключевым словам, непосредственно по тексту и с помощью содержания – хранятся в отдельных файлах, каждый из которых имеет свой HTML-подобный формат.

В этом большое неудобство данного формата документации. К тому же у CHM проблемы с кодировками: например, одна часть проекта (hhk – список ключевых слов для поиска) не поддерживает русский язык, его приходится заменять с помощью HTML-ной кодировки. Список становится нечитаемым. Другая часть проекта (hhc – содержание) русский язык поддерживает, но case-средство, которое официально поддерживает Microsoft (HelpWorkshop) заменяет все русские символы на всю ту же HTML-кодировку. Конечно, можно пользоваться исключительно этим case-средством и все эти технические проблемы оставить ему, но проблема в том, что само case-средство несовершенно: известна проблема, когда оно теряет некоторые библиотеки и уже не может компилировать документацию.

Сам Microsoft уже давно не развивает и не совершенствует свой формат: компания отказалась в начале 2000-х от него в пользу формата HLP, который начал применяться в новых версиях Windows.

При открытии документации автоматически вызывается средство просмотра, встроенное в Windows. HTML-код обрабатывается при помощи браузера Internet Explorer, который так же является разработкой Microsoft. Помимо того, что Microsoft уже не поддерживает данный формат, появляется проблема платформенной зависимости. Конечно, энтузиасты разработали средства просмотра для Linux-платформ, но формат документации все равно остается проприетарным, средства просмотра – полностью средства Microsoft. Поэтому эти решения неудовлетворительны.

В процессе компиляции все HTML-страницы перемещаются в корень архива, отсюда вытекает проблема со ссылками, т.е. все файлы должны находиться в одной директории. Никакой иерархии, только плоский список. Для хранения исходных данных такая концепция еще может быть приемлемой, но никак не для рабочей системы: именно поэтому невозможно сайт для документации РДО.

Подведем итог.

2.2.2. Достоинства существующей системы

- хранение документации в простом формате – HTML;
- возможность навигации по телу документации с помощью содержания и списка ключевых слов;
- полноценно представленная возможность формата HTML со стилями CSS и скриптами JavaScript;
- небольшое сжатие размеров файлов.

2.2.3. Недостатки существующей системы

- плоский список файлов HTML;
- платформенная зависимость документации – только Windows;
- неудобные форматы хранения данных для навигации по документации;
- плохо реализованный поиск непосредственно по телу документации;
- проблемы с кодировками;
- ненадежность case-средства;
- отсутствие возможности создания сайта документации.

2.2.4. Итог

Принято решение отказаться от формата CHM в виду его морального устаревания, а так же ряда технических проблем, ограничивающих возможности системы документации. Эти возможности уже не могут удовлетворить возросшие потребности пользователей и разработчиков РДО.

2.3. **Выбор системы документации**

Разумеется, при переходе от одной системы к другой есть непреодолимое желание сохранить все плюсы старой системы, получить новые плюсы и не получить минусы новой системы.

Была выбрана система документации Qt Assistant – компонент QT – кросс-платформенного инструментария разработки ПО на языке программирования C++.

Новая система сохраняет все плюсы старой системы и даже превосходит формат CHM по сжатию размеров документации: сжатие текста и изображений в 3-4 раза.

Кроме того, новая система обладает достоинствами, которые преимущественно пересекаются с недостатками старой:

- возможно произвольное расположение исходных файлов (в пределах текущей директории файла проекта документации);

- Qt – сама по себе кросс-платформенная система инструментов, поэтому исходные данные для документации созданные на одной платформе гарантируют работоспособность и на остальных платформах;

- формат проекта документации – xml-документ;

- хорошо реализованный поиск по телу документации;

- требование qt – работа в кодировке utf-8 – юникод;

- произвольное расположение файлов позволяет создать сайт документации РДО.

Недостатки системы документации на основе Qt Assistant:

- отсутствие case-средств для работы с проектом документации (проблема небольшая: используется формат xml – прост и понятен);

- необходимость в сопровождении документации программой Qt Assistant.

2.4. **Общие сведения о** Qt

Кросс-платформенный инструментарий разработки ПО на языке программирования C++. Есть также «привязки» ко многим другим языкам программирования: Python — PyQt, PySide; Ruby — QtRuby; Java — Qt Jambi; PHP — PHP-Qt и другие.

Позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Существуют версии библиотеки для Microsoft Windows, систем класса UNIX с графической подсистемой X11, iOS, Android, Mac OS X, Microsoft Windows CE, QNX[6], встраиваемых Linux-систем и платформы S60.

Qt распространяется по 3 лицензиям (независимо от лицензии, исходный код Qt один и тот же):

- Qt Commercial — для разработки ПО с собственнической лицензией, допускающая модификацию самой Qt без раскрытия изменений;
- GNU GPL — для разработки ПО с открытыми исходниками распространяемыми на условиях GNU GPL;
- GNU LGPL — для разработки ПО с собственнической лицензией, но без внесения изменений в Qt.

3. ТЕХНИЧЕСКОЕ ПРОЕКТИВРОВАНИЕ

3.1. **Планирование работ**

Стадии и этапы работ приведены в техническом задании (пункт 1.6). Ниже приведены пояснения принятых решений, а также общие сведения.

а) Прежде чем производить серьёзные изменения в системе, необходимо убедиться в том, что она сама по себе валидна. Неразумно ожидать, что из неработающей системы одного типа получится система другого типа, но уже рабочая. К тому же это необходимо для отслеживания процесса перехода: важно знать, появилась проблема в результате конвертации или существовала ранее.

б) Реализовано в пункте 2.3.

в) Выбор новой системы оказался довольно удачным: сам исходный текст документации обоих систем хранится в HTML-формате. Разница во вспомогательных данных: содержание и список ключевых слов. К сожалению, конвертеров одного формата в другой не существует. Разрабатывать такой конвертер для одного-единственного перехода – слишком накладно. Объем работ сопоставим с полностью ручным копированием содержания и списка ключевых слов. В результате было принято решение разбить эти списки на части (списки чрезвычайно не однородны по составу) и использовать регулярные выражения для обработки этих частей. Часть пришлось все же обработать вручную.

Требованием системы документации Qt является переход к кодировке UNICODE. Это положительная тенденция: повсеместное использование UNICODE позволит человечеству избавиться от проблем с кодировками, коих народилось за полвека достаточно, чтобы это проблемой.

К тому же было принято решение избавиться от разношерстности форматов хранимых изображений и перейти к единому формату: PNG.

г) После того, как состоится переход к новой системе, исчезнет необходимость хранения файлов плоским списком. Теперь можно менять организацию файлов: группировать их по логическим связям и выставлять зависимости между ними. Это влечет за собой правку связей между файлами.

д) Когда система ведения документации стала удобной, уместно править тело документации.

е) После того, как хранение исходных данных для документации станет содержать иерархию, будет возможно создать сайт документации. Сам текст документации хранится в формате HTML – основном формате, с которым работают все браузеры. Проблема возникает только с содержанием и списком ключевых слов. Для них нужно создать свои страницы, чтобы дать пользователю возможность читать документацию через сеть Интернет. Для создания страниц будет использован JavaScript, который создает само дерево с помощью особым образом форматированных строк, содержащие всю необходимую информацию. Эти строки имеют схожую структуру с форматом хранения проекта документации в XML.

ж) Благодаря сходствам из пункта (е) станет возможна разработка относительно несложного ПО для сопровождения сайта. Фактически с помощью

этого ПО (скрипта), написанного на языке PERL, будет заново формироваться сайт на основе файлов проекта документации Qt. Таким образом, изменения необходимо вносить только в системе документации, на сайте (который является отдельной системой) изменения появятся после использования скрипта.

з) Существует ПО для получения сайта. Существует система контроля версий SVN, которая позволяет удаленно хранить исходные данные и выгружать эти данные по запросу. Это позволяет организовать систему, которая будет самостоятельно обновлять сайт. Разработчикам РДО и документации к нему можно забыть про сайт. Изменения, внесенные разработчиком, появятся на сайте «сами».

и) Существование самой документации полезно, но еще полезнее возможность для пользователя РДО получить справку по конкретному элементу документации, который вызывает у пользователя затруднения. Это будет контекстно-чувствительный вызов, который необходимо разработать.

3.2. **Структура проекта документации**

Реальные данные документации, т.е. оглавление, предметный указатель или html-документы, содержатся в сжатых файлах документации Qt. Таким образом, один файл представляет, обычно, одно руководство или комплект документации. Так как большинство продуктов, являются достаточно полными и состоят из различных инструментов, то одно руководство используется достаточно редко. Вместо этого существует множество руководств, которые должны быть доступны одновременно. В идеале, также должна существовать возможность ссылаться на определенные интересные места из одного руководства в другое. Следовательно, справочная система Qt работает с файлами коллекции документации, которые включают любое количество сжатых файлов документации.

Однако наличие файлов коллекции для объединения множества комплектов документации может привести к некоторым проблемам. Например, одно ключевое слово предметного указателя может быть определено в различных документациях. Таким образом, только когда вы видите её в предметном указателе и выбираете, вы не можете быть уверены в том, что будет отображена ожидаемая документация. Следовательно, справочная система Qt предоставляет возможность фильтровать содержимое документации после определения свойств фильтра. Однако это требует, чтобы свойства были ассоциированы с содержимым документации перед генерацией сжатого файла документации.

Как уже упоминалось, сжатые файлы документации Qt содержат все данные, таким образом нет никакой необходимости поставлять всё едиными html-файлами. Вместо этого, необходимо распространять только сжатые файлы документации и, возможно, файл коллекции. Файл коллекции необязателен т.к. могут быть использованы любые существующие файлы коллекции, например, из строго выпуска.

Существуют четыре файла взаимодействующие со справочной системой, два используются для создания документации Qt и два предназначены для распространения:

- Проект документации Qt (Qt Help Project) – имеет расширение (.qhp). Исходный файл генератора документации, состоящий из таблицы оглавления, индексов и ссылок на реальные файлы документации (*.html); также этот файл определяет уникальное пространство имён для документации, на которую он ссылается.

- Сжатая справка Qt (Qt Compressed Help) – имеет расширение (.qch). Выходной файл генератора документации. Это двоичный файл, содержащий всю информацию, расположенную в файле проекта-документации, вместе со сжатыми файлами документации.

- Проект коллекции документации Qt (Qt Help Collection Project) – имеет расширение (.qhcp). Исходный файл для генератора коллекции документации. Содержит ссылки на сжатые файлы документации, которые нужно включить в коллекцию; также может содержать информацию о пользовательских настройках (customizing) программы Qt Assistant.

- Коллекция документации Qt (Qt Help Collection) – имеет расширение (.qhc). Выходной файл генератора коллекции. Это файл, с которым взаимодействует класс QHelpEngine. Содержит ссылки на любое количество сжатых файлов документации и дополнительную информацию, например пользовательские фильтры.

3.2.1. Создание проекта документации Qt

Сборка файлов документации для справочной системы Qt предполагает, что html-файлы документации уже существуют, т.е. справочная система Qt не предоставляет возможности создания html-файлов, как например, Doxygen.

Если html-документы присутствуют, файл проекта документации Qt должен быть создан. После указания всей необходимой информации в этом файле, его необходимо скомпилировать вызвав:

```
qhelpgenerator doc.qhp -o doc.qch
```

Файл 'doc.qch' будет содержать все html-файлы в сжатой форме вместе с оглавлением и предметным указателем. Чтобы проверить, является ли созданный файл правильным, откройте Qt Assistant и установите файл через страницу Settings|Documentation.

3.2.1.1. Проект документации Qt

Проект документации Qt объединяет все данные необходимые для создания сжатых файлов документации. Вместе с реальными данными документации, такими как оглавление, индекс ключевых слов и справочные документы, он содержит некоторую дополнительную информацию такую как пространства имен, чтобы

идентифицировать файл документации. Один проект документации определяет одну документацию, например, руководство по Qt Assistant.

3.2.1.2. Формат проекта документации Qt

Формат файла основан на XML. Для лучшего понимания формата мы обсудим следующий пример

```
<?xml version="1.0" encoding="UTF-8"?>
 <QtHelpProject version="1.0">
    <namespace>mycompany.com.myapplication.1.0</namespace>
    <virtualFolder>doc</virtualFolder>
    <customFilter name="Моё приложение 1.0">
       <filterAttribute>myapp</filterAttribute>
       <filterAttribute>1.0</filterAttribute>
    </customFilter>
    <filterSection>
       <filterAttribute>myapp</filterAttribute>
       <filterAttribute>1.0</filterAttribute>
       <toc>
          <section title="Руководство по моему приложению" ref="index.html">
             <section title="Глава 1" ref="doc.html#chapter1"/>
             <section title="Глава 2" ref="doc.html#chapter2"/>
             <section title="Глава 3" ref="doc.html#chapter3"/>
          </section>
       </toc>
       <keywords>
          <keyword name="foo" id="MyApplication::foo" ref="doc.html#foo"/>
          <keyword name="bar" ref="doc.html#bar"/>
          <keyword id="MyApplication::foobar" ref="doc.html#foobar"/>
       </keywords>
       <files>
          <file>classic.css</file>
          <file>*.html</file>
       </files>
    </filterSection>
 </QtHelpProject>
```

3.2.1.3. Пространства имён

Чтобы дать возможность QHelpEngine находить соответствующую документацию по заданной ссылке, каждый набор документации должен иметь уникальный идентификатор. Уникальный идентификатор также дает возможность, для коллекции документации, отследить набор документации, не ссылаясь на его имя файла. Справочная система Qt использует пространства имен, как

идентификатор, который определяется обязательным тегом пространства имен. В примере выше, пространством имен является "mycompany.com.myapplication.1.0".

3.2.1.4. Виртуальные каталоги

Наличие пространства имен для каждой документации, естественно означает, что наборы документации совершенно независимы. С точки зрения движка документации это благоприятно, но с точки зрения документатора часто желательно, чтобы были перекрестные ссылки определенной темы из одного руководства в другое, без необходимости задания абсолютных ссылок. Чтобы решить эту проблему, справочная система вводит концепцию виртуальных каталогов.

Виртуальный каталог становится корневым каталогом всех файлов, указанных в сжатом файле документации. Когда две документации разделяют между собой один и тот же виртуальный каталог, они могут использовать относительные пути, определяя гиперссылки, указывающие на другую документацию. Если файл содержится в обеих документациях или руководствах, файл из текущего руководства имеет приоритет над другим.

```
...
<virtualFolder>doc</virtualFolder>
...
```

Пример выше устанавливает 'doc' как виртуальный каталог. Если другое руководство, например, для маленького вспомогательного инструмента к 'Моему приложению' указывает тот же каталог, достаточно написать 'doc.html#section1', чтобы сослаться на первый раздел в руководстве 'Моё приложение'.

Тег виртуального каталога является обязательным и каталог не должен содержать символы '/'.

3.2.1.5. Пользовательские фильтры

Следующее, в файле проекта Qt - это необязательное определение пользовательских фильтров. Пользовательский фильтр содержит список свойств фильтра, которые будут использованы позднее, чтобы отобразить только ту документацию, которая имеет все эти свойства. Таким образом, при установке текущего фильтра в QHelpEngine в значение "Моё приложение 1.0" будет отображена только та документация, которая имеет в качестве свойств фильтра "MyApp" и "1.0".

```
...
<customFilter name="Моё приложение 1.0">
    <filterAttribute>myapp</filterAttribute>
    <filterAttribute>1.0</filterAttribute>
</customFilter>
...
```

Можно определить любое количество пользовательских фильтров в файле проекта документации. Важно знать, что свойства фильтра не обязательно указывать в этом же файле проекта; они могут быть определены в любом другом файле документации. Определение свойств фильтра осуществляется путем указания их в разделе фильтра.

3.2.1.6. Раздел фильтра

Раздел фильтра содержит реальную документацию. Один файл проекта документации Qt может содержать более одного раздела фильтра. Каждый раздел фильтра состоит из четырех частей, раздел свойств фильтра, оглавление, ключевые слова и список файлов. Теоретически все части необязательны, но не определение хотя бы одной из них приведет к пустой документации.

Свойства фильтра

Каждый раздел фильтра должен иметь свойства, ассоциированные с фильтром, чтобы обеспечить фильтрование документации. Если свойства фильтра не определены, документация будет отображена только, если не происходит фильтрация, то есть текущий пользовательский фильтр в QHelpEngine не содержит каких-либо свойств фильтра.

```
...
 <filterSection>
    <filterAttribute>myapp</filterAttribute>
    <filterAttribute>1.0</filterAttribute>
 ...
```

В этом случае, свойства фильтра 'myapp' и '1.0' указаны в разделе фильтра, т.е. всё содержимое, определенное в этом разделе, будет отображено только, если текущий пользовательский фильтр имеет, в качестве свойств фильтра, или 'myapp' или '1.0' или оба вместе.

Оглавление

```
...
 <toc>
    <section title="Руководство по моему приложению" ref="index.html">
       <section title="Глава 1" ref="doc.html#chapter1"/>
       <section title="Глава 2" ref="doc.html#chapter2"/>
       <section title="Глава 3" ref="doc.html#chapter3"/>
    </section>
 </toc>
 ...
```

Один тег раздела (section) представляет одну запись в оглавлении. Разделы могут быть вложены в любой степени, но, исходя из удобств пользователей, они не должны иметь более четырёх-пяти уровней. Раздел определяется его заголовком и ссылкой. Ссылки, как и все ссылки на файлы, в проекте документации Qt задаются относительно самого файла проекта документации.

Замечание: Ссылочные файлы должны быть в том же каталоге (или внутри подкаталога) что и файл проекта документации. Абсолютный путь к файлу также не поддерживается.

Ключевые слова

```
...
<keywords>
   <keyword name="foo" id="MyApplication::foo" ref="doc.html#foo"/>
   <keyword name="bar" ref="doc.html#bar"/>
   <keyword id="MyApplication::foobar" ref="doc.html#foobar"/>
</keywords>
...
```

Раздел ключевых слов перечисляет все ключевые слова этого раздела фильтра. Ключевое слово состоит, в основном, из имени и ссылки на файл. Если атрибут 'name' используется, то ключевое слово, определенное в нем, появится в предметном указателе, т.е. оно будет доступным через QHelpIndexModel. Если 'id' используется, ключевое слово не появляется в предметном указателе и и доступно только через функцию linksForIdentifier() QHelpEngineCore. Атрибуты 'name' и 'id' могут быть определены одновременно.

Файлы

```
...
<files>
    <file>classic.css</file>
    <file>*.html</file>
</files>
...
```

В конце, должны быть перечислены реальные файлы документации. Убедитесь, что указаны все файлы, необходимые для отображения документации, т.е. таблицы стилей или подобные файлы необходимо указать здесь же. Файлы, как все ссылки на файлы в проекте документации Qt, относятся к самому файлу проекта документации. Как показано в примерах, файлы (но не каталоги) можно также задать как шаблоны используя специальные символы. Все перечисленные файлы будут сжаты и записаны в сжатый файл документации Qt (Qt compressed help file). Таким образом, в конце, один единственный файл документации Qt содержит все файлы

документации вместе с содержанием и предметным указателем. Замечание: Ссылочные файлы должны быть в том же каталоге (или внутри подкаталога) что и файл проекта документации. Абсолютный путь к файлу также не поддерживается.

3.2.2. Создание коллекции документации Qt

Первый шаг - создание файла проекта коллекции документации Qt. Так как коллекция документации Qt хранит, главным образом, ссылки на сжатые файлы документации, то файл проекта 'mycollection.qhcp' выглядит неудивительно простым:

```
<?xml version="1.0" encoding="utf-8" ?>
 <QHelpCollectionProject version="1.0">
    <docFiles>
       <register>
          <file>doc.qch</file>
       </register>
    </docFiles>
 </QHelpCollectionProject>
```

Для того, чтобы фактически создать файл коллекции, нужно вызвать:

qcollectiongenerator mycollection.qhcp -o mycollection.qhc

Вместо запуска двух инструментов, один для создания сжатой документации, а другой для создания файла коллекции, также возможно просто запустить инструмент qcollectiongenerator со слегка изменённым файлом проекта, информируя генератор о том, чтобы необходимо сначала создать сжатую справку.

```
...
 <docFiles>
    <generate>
       <file>
          <input>doc.qhp</input>
          <output>doc.qch</output>
       </file>
    </generate>
    <register>
       <file>doc.qch</file>
    </register>
 </docFiles>
 ...
```

Конечно, можно определить более одного файла в секции 'generate' или 'register', так что любое количество сжатых файлов документации может быть создано и зарегистрировано сразу.

3.2.3. Использование документации Qt

Доступ к содержимому документации возможен двумя способами: Используя Qt Assistant, как просмотрщика документации, или используя QHelpEngine API для встраивания содержимого документации прямо в приложение. В рамках данного курсового проекта решено использовать Qt Assistant.

3.2.4. Использование Qt Assistant

Qt Assistant оперирует файлом коллекции, который может быть задан перед запуском. Если файл коллекции не задан, то будет создан и использован файл по умолчанию. В другом случае, можно зарегистрировать любой сжатый файл документации Qt и получить доступ к содержимому документации.

Когда Assistant используется как просмотрщик документации для приложения, желательно, чтобы он мог быть настроен, чтобы лучше соответствовать приложению и не выглядеть как независимый, самодостаточный просмотрщик документации. Чтобы добиться этого, некоторые дополнительные свойства могут быть установлены в файле коллекции Qt, например, чтобы изменить заголовок или значок приложения Qt Assistant. Для получения дополнительной информации по этой теме смотрите Руководство по Qt Assistant.

3.3. **Скрипт для создания сайта**

Так как получение страниц сайта может быть достаточно четко формализовано, то возможно создавать автоматически. Например, с помощью скрипта.

Для разработки скрипта выбран язык Perl. Он обладает хорошим инструментарием для работы со строками. К сожалению, это его единственная сильная сторона, но с поставленной задачей он справится. К тому же он легок в изучении.

3.4. **Управление** Qt Assistant’**ом**

Реализуется с помощью библиотек Qt.

Хотя система документации - автономное приложение, в основном, она будет запускаться приложением, для которого она предоставляет справку. Этот подход дает приложению возможность, указывать определенное содержание документации для отображения при запуске справочной системы. Другое преимущество, связанное с этим подходом - то, что приложение может взаимодействовать с системой документации и, следовательно, может запрашивать другое содержимое документации, чтобы отображать его в зависимости от текущего состояния приложения.

Чтобы использовать Qt Assistant, как пользовательскую систему документации приложения, нужно создать QProcess и указать путь к исполняемому файлу Assistant.

Для того, чтобы Assistant слышал приложение, надо включить его функции дистанционного управления, передав аргумент командной строки - enableRemoteControl.

```
QProcess *process = new QProcess;
QStringList args;
args << QLatin1String("-collectionFile")
    << QLatin1String("mycollection.qhc")
    << QLatin1String("-enableRemoteControl");
process->start(QLatin1String("assistant"), args);
if (!process->waitForStarted())
    return;
```

Когда Qt Assistant запущен, можно отправлять команды, используя поток стандартного ввода (stdin) процесса. Фрагмент кода ниже, показывает, как сказать Qt Assistant, чтобы он показал определенную страницу в документации.

```
QByteArray ba;
ba.append("setSource qthelp://com.mycompany.1_0_0/doc/index.html\n");
process->write(ba);
```

4. РАБОЧЕЕ ПРОЕКТИРОВАНИЕ

4.1. **Обзор формата** CHM

Формат проекта документации Qt был описан в пункте 3.2.

Проект формата CHM в обязательном порядке должен содержать основной файл проекта (.hhp) и файл содержания (.hhc). Остальные файлы опциональны. Среди них в документации РДО использовался только список ключевых слов (.hhk). Так как осуществляется процесс ухода от этого формата, то не стану приводить описание всех остальных опций CHM.

Файл HHP - основной файл проекта документации. В нем указываются основные параметры окна документации, а также перечисляются все HTML-страницы, входящие в справку. Типичное содержимое файла HHP приведено ниже:

```
[OPTIONS]
Compatibility=1.1 or later
Compiled file=help.chm
Contents file=help.hhc
Default window=main
Default topic=index.php
Error log file=Errlog.txt
Full-text search=Yes
Index file=help.hhk
Language=0x419 Russian
[WINDOWS]
main="Title","help.hhc","help.hhk","index.php","home.php",,,,,131104,200,8198,[119,78,769,534],0,0,,,,1,0
[FILES]
C:\page1.php
C:\page2.php
```

Compatibility=... - определяет совместимость полученного файла документации с программой просмотра (HH.EXE). Если Вы не хотите получить лишние проблемы, указывайте именно версию 1.1.

Compiled file=... - имя итогового файла формата chm.

Contents file=... - имя файла содержания (формат рассматривается ниже).

Default window=... - название окна, в котором откроется справка во время просмотра. Желательно задать имя "main".

Default topic=... - страница документации, которая будет открываться сразу после открытия.

Error log file=... - имя файла, в который будут сохраняться сообщения, выдаваемые программой при компиляции проекта. Если этот файл не нужен, просто удалите данную строку.

Full-text search=Yes - параметр, определяющий отображение вкладки "Поиск" и, соответственно, возможность полнотекстового поиска по файлу документации.

Index file=... - параметр определяет имя файла Индекса. Параметр указывается в случае наличия вкладки Индекс и/или Поиск.

Language=... - язык компиляции. Для проектов, содержащих русские названия страниц, НЕОБХОДИМО указывать Русский.

main=... - это самый важный параметр файла HHP. Именно этот параметр задает размеры, стиль и расположение окна просмотра. Если в параметре "Default window" вы задали другое имя окна, измените название этого параметра на соответствующее значение.

[FILES] - в этой секции указываются названия всех файлов HTML-страниц, входящих в файл документации.

Параметры main:

"Title" - название справочной системы, которое отображается в заголовке окна просмотра документации.

"help.hhc" - имя файла Содержания документации.

"help.hhk" - имя файла Индекса документации.

"index.php" - имя страницы по умолчанию - открывается сразу после загрузки файла.

"home.php" - имя домашней страницы - открывается при нажатии кнопки "Домой" панели инструментов программы просмотра.

Следующие 4 параметра предназначены для определения двух пользовательских кнопок, которые будут отображаться на панели инструментов, например: ,"about.php","о проекте","help.php","помощь",

131104 - определяет количество вкладок. Если этот параметр равен 132128, то отображаются вкладки Содержание, Поиск и/или Индекс (в зависимости от указанных выше параметров). Если 131104, то отображаются вкладки Содержание и (если указано выше) Индекс. Чтобы не выводить текст на кнопках панели инструментов, прибавьте к значению этого параметра 64.

200 - ширина навигационной панели в пикселах.

8198 - количество и вид кнопок панели инструментов.

[119,78,769,534] - соответственно положение и размер окна документации в пикселах.

0 - нет данных.

0 - внешний вид окна. Если требуется трехмерная рамка вокруг окна, то значение этого параметра 512.

Следующие 3 параметра - назначение неизвестно.

1 - местоположение вкладок на панели навигации. "0" - стандартное; "1" - слева; "2" - снизу.

0 - нет данных.

Файл HHC - это файл Содержания документации. Пример типичного содержимого файла HHC приведен ниже.

```
<!DOCTYPE HTML PUBLIC "-//IETF/DTD HTML/EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="I and my Hands">
<!-- Sitemap 1.0 -->
</HEAD><BODY>
<OBJECT type="text/site properties">
<param name="FrameName" value="right">
<param name="ImageType" value="Folder">
<param name="Window Styles" value="0x27">
<param name="Foreground" value="0x80000005">
<param name="Background" value="0x80000005">
<param name="Font" value="MS Sans Serif,8,0">
</OBJECT>
<UL>
<LI> <OBJECT type="text/sitemap">
<param name="Name" value="О программе">
<param name="Local" value="about.php">
<param name="ImageNumber" value="12">
</OBJECT>
<UL>
<LI> <OBJECT type="text/sitemap">
<param name="Name" value="Меню программы">
<param name="Local" value="menu.php">
<param name="ImageNumber" value="11">
</OBJECT>
</UL>
</UL>
</BODY></HTML>
```

Как видите, структура файла очень напоминает структуру обычного HTML-документа. Первая строка определяет схему разметки документа. Далее идут строки, стандартные для любого HTML-файла. Особое внимание следует обратить на параметр "Window Styles". Его значение определяет стиль окна и панели навигации программы просмотра.

Каждый заголовок раздела в этом файле представлен в виде тегов <object></object>, между которыми заключаются параметры, определяющие имя, название файла и иконку данного раздела в панели навигации. Все имена разделов заключены в список (тегами). Если какой-либо раздел имеет подразделы, то они заключаются в отдельный список.

Файл HHK - создается, если необходимо отобразить вкладку Индекс панели навигации. Пример типичного содержимого файла Индекса представлен ниже.

```
<!DOCTYPE HTML PUBLIC "-//IETF/DTD HTML/EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="I and my Hands">
<!-- Sitemap 1.0 -->
</HEAD><BODY>
<OBJECT type="text/site properties">
</OBJECT>
<UL>
<LI> <OBJECT type="text/sitemap">
<param name="Keyword" value="О программе">
<param name="Local" value="about.php">
</OBJECT>
<LI> <OBJECT type="text/sitemap">
<param name="Keyword" value="Меню программы">
<param name="Local" value="menu.php">
</OBJECT>
</UL>
</BODY></HTML>
```

Структура этого файла аналогична структуре HHC и также представляет собой HTML-документ.

В первой строке определяется структура файла, далее идут обязательные для любого HTML-документа теги, за которыми следует пустой тег <object></object>. Далее начинается список, содержащий все элементы индекса. Так как индекс не имеет иерархии, то никаких вложенных списков не допускается.

4.2. **Переход к новому формату**

Два блока документации rdo_lang_rus и rdo_studio_rus решено объединить в один: они будут логически разделены внутри самого проекта документации.

Содержимое обоих файлов hhc должно быть переформатировано и добавлено в раздел < toc ></ toc >. Содержимое обоих файлов hhk должно быть переформатировано и добавлено в раздел <keywords></keywords>.

Для однородных частей этих файлов возможно применить регулярные выражения. К тому же необходимо изменить кодировку файлов на UNICODE. Текстовый редактор Notepad++ прекрасно работает и с регулярными выражениями, и с конвертированием файлов. Поэтому он и был выбран.

4.3. **Рефакторинг**

Исходные файлы тела документации решено сгруппировать по разделам документации. Например, файлы:

rdo_base_alpha.htm
rdo_base_equations.htm
rdo_base_instr.htm
rdo_base_intro.htm
rdo_base_proc_lang.htm
rdo_base_resources.htm
rdo_base_res_types.htm
rdo_base_std_var_func.htm
rdo_base_types.htm

теперь находятся в директории rdo_base. Аналогично – все остальные файлы.

Простая операция порождает непростую задачу: после перемещения изменилось относительное расположение файлов этого раздела и всех остальных, и это значит, что старые связи между ними (ссылки) больше не работают. Перекрестных ссылок много: порядка 200. Почти все из них нужно откорректировать (не корректируются только ссылки из этого же раздела) и обязательно проверить (открыть страницу, найти эту ссылку, убедиться в работоспособности). Очень трудоемкое занятие. Принято решение использовать регулярные выражения для коррекции ссылок. Увы, не удалось шаблонным решением получить идеальный результат: некоторые ссылки оказались нерабочими. Для этого было использовано ПО Web Link Validator – валидатор ссылок. Это ПО находит и отображает нерабочие ссылки, которые корректировались вручную.

4.4. **Коррекция тела документации**

Все изменения в теле документации описывать нет смысла. Продемонстрирую самый яркий пример модификации.

Раздел «Описание образцов» содержал слишком много разрозненных данных, например, «тело_образца» содержало уточнения для каждого из видов образцов в то время как для них существуют свои страницы. «Тело_образца» было вынесено на каждую из страниц. Но тогда получается слишком много повторяющего текста: «инструкции» - понятие, которое встречается не только в теле образца, поэтому решено его перенести в раздел «Процедурное программирование». Не смотря на то, что инструкции относятся к теме процедурного программирования, их нельзя объединять в общий раздел с операторами. Поэтому инструкции вынесены в отдельный раздел.

К тому же раздел «Описание образцов» содержит объемные примеры, которые решено скрыть и показывать по желанию пользователя.

В результате произведенных манипуляций раздел стал гораздо комфортнее читаем. Аналогичные манипуляции были произведены с разделами:

- Последовательности;
- Кадры анимации;

- Процедурное программирование;
- Объект описания показателей;
- Объект прогона;
- Процессы обслуживания;
- Точки принятия решений;
- Функции;

4.5. **Создание прототипа сайта на основе дерева**

Любая объемная документация должна иметь содержание, с помощью которого можно легко и просто перейти на нужную страницу. Значит, и сайт должен иметь содержание. Но начинающий пользователь скорее всего будет плохо ориентироваться в терминологии РДО, и у него будут возникать вопросы типа «Что значит это слово?». В справке CHM и Qt ему в этом поможет список ключевых слов. Значит, и на сайте он должен быть. Для поиска по всему тексту можно воспользоваться таким онлайн сервисом как Google. Итак, содержание и список ключевых слов.

Оба списка решено сделать в виде дерева. Дизайн сайта выбран минималистический. В общем и целом соответствует дизайну Qt Assisstant.

При создании сайта используются элементы HTML5 – последней версии этого стандарта. Также используются стили CSS.

Само дерево генерируется с помощью JavaScript. Для него необходимы особым образом форматированные строки, похожие на строки в проекте документации Qt. Это можно реализовать с помощью регулярных выражений, а также MS Excell. Неудобно, но для прототипа достаточно.

В результате есть концепция двух страниц, имеющих ссылки друг на друга.

4.6. **Написание скриптов для создания сайта**

Для двух страниц нужно создать два скрипта: для содержания и для списка ключевых слов.

Суть скриптов сводится к тому, что из файлов проекта документации Qt вычленяются соответствующие строки, форматируются нужным образом, индексируются и добавляются заголовок и концевик HTML-страницы. Есть особенности для каждого скрипта.

Скрипт для содержания:

Нужно последовательно выбрать данные сначала из одного проекта, затем из другого. При форматировании строк нужно выстроить иерархию. Она будет выстраиваться на основе синтаксиса строк. Скрипт хранится в файле help-index.pl (см. Приложение 1).

Скрипт для списка ключевых слов:

Нужно выбрать данные из обоих проектов, далее отсортировать по алфавиту. Сам по себе список ключевых слов – плоский. Но существуют ключевые слова, которые ведут на несколько страниц. В проекте Qt их группирует сам Qt Assistant. Для сайта должен сгруппировать скрипт. Скрипт хранится в файле keyword-index.pl (см. Приложение 2).

4.7. **Настройка автообновления сайта**

Так как Perl-скрипты – исполняемые файлы, то настройка автообновления не представляет большой проблемы. Это будет организовано с помощью BAT-скрипта и Планировщика Windows. По расписанию (каждую ночь) будет выгружаться текущая версия документации, на ее основе создаваться новый сайт и замещать старый.

Код скрипта хранится в файле updater.bat (см. Приложении 3).

4.8. **Контекстный вызов документации**

Пользователь должен иметь возможность вызвать нужный раздел документации в разделе «Опции», при фокусировке на окнах «Трассировка», «Вывод», «Компилятор», «Результаты», «Поиск», «Графики», «Анимация». Также нужна особенная реакция на вызов документации при выделении ключевых слов.

Код, реализующий эти возможности, представлен в приложении 4.

СПИСОК ЛИТЕРАТУРЫ

1) Справка РДО [В Интернете]
http://rdo.rk9.bmstu.ru/forum/download/file.php?id=4
2) Сведения о формате CHM [В Интернете]
http://ru.wikipedia.org/wiki/Microsoft_Compiled_HTML_Help
3) Сведения о Qt [В Интернете]
http://ru.wikipedia.org/wiki/Qt
4) Сведения о каркасе документации Qt [В Интернете]
http://doc.crossplatform.ru/qt/4.7.x/qthelp-framework.html
5) Управление Qt Assistant [В Интернете]
http://doc.crossplatform.ru/qt/4.7.x/assistant-custom-help-viewer.html
6) Структура проекта документации Qt [В Интернете]
http://www.doc.crossplatform.ru/qt/4.7.x//qthelpproject.html
7) Прототип дерева для сайта [В Интернете]
www.destroydrop.com/hjavascripts/tree/

СПИСОК ПРИМЕНЯЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1. Microsoft Office 2010
2. Microsoft Visual Studio 2008
3. Microsoft Paint 6.1
4. Notepad++ v6.1
5. Web Link Validator
6. ActivePerl-5.14.2.1402
7. dwimperl-5.14.2.1-v7
8. Qt Assistant
9. qt-everywhere-opensource-src-4.8.1

ПРИЛОЖЕНИЕ 1. Код скрипта help-index.pl

```
use strict;

open(source_studio,"rdo_studio_rus.qhp");
open(source_lang,"rdo_lang_rus.qhp");

if (-e "help-index.html") {
	unlink ("help-index.html");
}
open(target,">>help-index.html");

my @temp;
my $flagO=0;
my $flagC=0;
my @strings_studio=<source_studio>;
foreach my $line (@strings_studio)
{
	if ($line=~ /<\/toc>/) { $flagC = 1; }
	if ( $flagO != $flagC ) {
		push @temp, $line;
	}
	if ($line=~ /<toc>/) { $flagO = 1; }
}


my $flagO=0;
my $flagC=0;
my @strings_lang=<source_lang>;
foreach my $line (@strings_lang)
{
	if ($line=~ /<\/toc>/) { $flagC = 1; }
	if ( $flagO != $flagC ) {
		push @temp, $line;
	}
	if ($line=~ /<toc>/) { $flagO = 1; }
}
close (source_studio);
close (source_lang);


my $it = 0;
my $ii = 1;
my $size = @temp;
my @parents;
```

```
push @parents, $it;
foreach my $line (@temp)
{
	if ( $line =~ /\"\/>/ ) {
		$line =~ s/<section title="/Tree[$it] = \"$ii\|$parents[-1]\|/;
		$line =~ s/\" ref=\"/\|help\//;
		$line =~ s/\"\/>/\";/;

		++$it;
		++$ii;
	}
	else {if ( $line =~ /\">/ ) {
		$line =~ s/<section title="/Tree[$it] = \"$ii\|$parents[-1]\|/;
		$line =~ s/\" ref=\"/\|help\//;
		$line =~ s/\">/\";/;

		push @parents, $ii;
		++$it;
		++$ii;
	}
	else  {
		$line =~ s/\t+<\/section>\n//;
		pop @parents;
	}}

}

open(source_html_head,"help-index-head.tmp");
my @strings_html_head=<source_html_head>;
foreach my $line (@strings_html_head)
{
	print (target $line);
}
close (source_html_head);

print (target @temp);

open(source_html_body,"help-index-body.tmp");
my @strings_html_body=<source_html_body>;
foreach my $line (@strings_html_body)
{
	print (target $line);
}
close (source_html_body);
```

ПРИЛОЖЕНИЕ 2. Код скрипта keyword-index.pl

```
use strict;

open(source_studio,"rdo_studio_rus.qhp");
open(source_lang,"rdo_lang_rus.qhp");

if (-e "keyword-index.html") {
    unlink ("keyword-index.html");
}
open(target,">>keyword-index.html");

my @temp;
my @strings_studio=<source_studio>;
foreach my $line (@strings_studio)
{
    if ($line=~ /keyword name=/) {
            push @temp, $line;
    }
}

my @strings_lang=<source_lang>;
foreach my $line (@strings_lang)
{
    if ($line=~ /keyword name=/) {
            push @temp, $line;
    }
}
close (source_studio);
close (source_lang);

my @temp2 = sort { uc (pack 'U0C*', unpack 'C*', $a) cmp uc (pack 'U0C*', unpack 'C*', $b) }
@temp;
my @temp3;
foreach my $line (@temp2)
{
    $line =~ s/\t{3}<keyword name=\"//;
    $line =~ s/\" ref=\"/|/;
    $line =~ s/\"\/>\n//;
    my @a = split(/\|/, $line);
    push @temp3, ([$a[0],$a[1]]);
}
```

```
sub callerUrl {
    my $path = $_[0];
    open(tempFile,$path);
    my @stringsTemp=<tempFile>;
    foreach my $line (@stringsTemp)
    {
        if ($line =~ /<TITLE>/) {
            $line =~ s/<TITLE>//;
            $line =~ s/<\/TITLE>\n//;
            return $line;
        }
    }
    close(tempFile);
}

open(source_html_head,"keyword-index-head.tmp");
my @strings3=<source_html_head>;
foreach my $line (@strings3)
{
    print (target $line);
}
close (source_html_head);

my $it = 0;
my $ii = 1;
my $size = @temp3;
for (my $i = 0; $i <= $size-1; ++$i,++$ii,++$it)
{
    if ($temp3[$i][0] eq $temp3[$i+1][0]) { #печать сгруппированных строк
        my $parrent = $ii;
        print (target "\t\tList[$it] = \"$ii\|0\|$temp3[$i][0]\|\";\n"); #печать строки-родителя
        --$i;
        do  {
            ++$i;
            ++$ii;
            ++$it;
            my $parrentName = callerUrl ($temp3[$i][1]);
            print (target "\t\t\tList[$it] =
\"$ii\|$parrent\|$parrentName\|help\/$temp3[$i][1]\";\n"); #печать строки-потомка
        } until ($temp3[$i][0] ne $temp3[$i+1][0])
    }
    else {
        if ( $temp3[$i][0] eq '\\' ) { print (target "\t\tList[$it] = \"$ii\|0\|\\\\\|help\/$temp3[$i][1]\";\n");
}
        else { print (target "\t\tList[$it] = \"$ii\|0\|$temp3[$i][0]\|help\/$temp3[$i][1]\";\n"); }
```

```
    }
}

open(source_html_body,"keyword-index-body.tmp");
my @strings4=<source_html_body>;
foreach my $line (@strings4)
{
    print (target $line);
}
close (source_html_body);
close (target);
```

ПРИЛОЖЕНИЕ 3. Код скрипта updater.bat

```
@echo off
set PATH1=D:\Dev_web
set PATH2=D:\Dev\app\rdo_studio_mfc\help
set PATH3=D:\Dev_web\web\scripts\Perl

TortoiseProc.exe /command:cleanup /PATH:"%PATH1%" /nodlg /externals /cleanup  /closeonend:1
/noui
TortoiseProc.exe /command:update /PATH:"%PATH1%" /closeonend:1

TortoiseProc.exe /command:cleanup /PATH:"%PATH2%" /nodlg /externals /cleanup  /closeonend:1
/noui
TortoiseProc.exe /command:update /PATH:"%PATH2%" /closeonend:1

cd %PATH3%
copy /Y help-index-body.tmp %PATH2%
copy /Y help-index-head.tmp %PATH2%
copy /Y help-index.pl %PATH2%
copy /Y keyword-index-body.tmp %PATH2%
copy /Y keyword-index-head.tmp %PATH2%
copy /Y keyword-index.pl %PATH2%

cd %PATH2%
help-index.pl
keyword-index.pl
copy /Y keyword-index.html %PATH1%
copy /Y help-index.html %PATH1%
```

options.cpp

```
...
	void RDOStudioOptions::onHelpButton()
	{
		QProcess* assistant = studioApp.chkQtAssistantWindow();
		if ( assistant->state() != assistant->Running ) return;
		QByteArray ba;

		CPropertyPage* page = GetActivePage( );
		if ( page == general ) {
			ba.append("setSource
	qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_general.htm\n");
		} else if ( page == editor ) {
			ba.append("setSource
	qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm\n");
		} else if ( page == tabs ) {
			ba.append("setSource
	qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_tabs.htm\n");
		} else if ( page == styles ) {
			ba.append("setSource
	qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_styles_and_color.htm\n");
		} else if ( page == plugins ) {
			ba.append("setSource
	qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options.htm\n");
		}
		assistant->write(ba);
	}

	BOOL RDOStudioOptions::OnHelpInfo(PTR(HELPINFO) pHelpInfo)
	{
		QProcess* assistant = studioApp.chkQtAssistantWindow();
		if ( assistant->state() != assistant->Running ) return TRUE;

		QByteArray ba;
		ba.append("setSource ");
		ba.append(resolveKeyAndUrl(pHelpInfo->dwContextId).c_str());
		ba.append("\n");
		assistant->write(ba);
		return FALSE;
	}
```

```
tstring RDOStudioOptions::resolveKeyAndUrl (ruint helpInfo)
{
	if (m_keyAndUrl.size() == 0)
		buildMap();
	mapKeyAndUrl::iterator it = m_keyAndUrl.find(helpInfo);
	return (*it).second;
}

void RDOStudioOptions::buildMap()
{
	m_keyAndUrl.insert
(std::pair<ruint,tstring>(0x80980413,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm#buffers"));
	m_keyAndUrl.insert
(std::pair<ruint,tstring>(0x809803e9,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm#clear_auto"));
	m_keyAndUrl.insert
(std::pair<ruint,tstring>(0x80980414,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm#clear_after"));
	m_keyAndUrl.insert
(std::pair<ruint,tstring>(0x80980415,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm#clear_after"));
	m_keyAndUrl.insert
(std::pair<ruint,tstring>(0x80980416,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm#clear_after"));
	m_keyAndUrl.insert
(std::pair<ruint,tstring>(0x8098040f,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm#complete"));
	m_keyAndUrl.insert
(std::pair<ruint,tstring>(0x809803fe,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm#full_nearest"));
	m_keyAndUrl.insert
(std::pair<ruint,tstring>(0x809803ff,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm#full_nearest"));
	m_keyAndUrl.insert
(std::pair<ruint,tstring>(0x809803fd,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_editor.htm#use_autocomlete"));
…

…
(std::pair<ruint,tstring>(0x809903f5,"qthelp://studio/doc/rdo_studio_rus/html/work_options/work_options_styles_and_color.htm#preview_as"));
}
```

rdoeditoredit.cpp

```
void RDOEditorEdit::OnHelpKeyword()
{
	QProcess* assistant = studioApp.chkQtAssistantWindow();
	if ( assistant->state() != assistant->Running ) return;

	tstring keyword = getCurrentOrSelectedWord();
	tstring s = getAllKW();

	if ( s.find_first_of( keyword ) == tstring::npos || keyword.empty() ) {
		RDOEditorTabCtrl* tab = model->getTab();
		if ( tab ) {
			switch( tab->getCurrentRDOItem() ) {
				case rdoModelObjects::RTP: keyword = "rtp"; break;
				case rdoModelObjects::RSS: keyword = "rss"; break;
				case rdoModelObjects::EVN: keyword = "evn"; break;
				case rdoModelObjects::PAT: keyword = "pat"; break;
				case rdoModelObjects::DPT: keyword = "dpt"; break;
				case rdoModelObjects::PRC: keyword = "prc"; break;
				case rdoModelObjects::FRM: keyword = "frm"; break;
				case rdoModelObjects::FUN: keyword = "fun"; break;
				case rdoModelObjects::SMR: keyword = "smr"; break;
				case rdoModelObjects::PMD: keyword = "pmd"; break;
				default:                   keyword = ""; break;
			}
		}
	}

	QByteArray ba;
	ba.append("activateKeyword ");
	ba.append(keyword.c_str());
	ba.append("\n");
	assistant->write(ba);
}
```

application.cpp

```
tstring RDOStudioApp::getFullHelpFileName(tstring str) const
{
	tstring strTemp = chkHelpExist(str);
	if (                    strTemp.size() < 3) return _T("");
	if (chkHelpExist ("assistant.exe").size() < 3) return _T("");
	return strTemp;
}

tstring RDOStudioApp::chkHelpExist(tstring fileName) const
{
	fileName.insert(0, rdo::extractFilePath(RDOStudioApp::getFullExtName()));
	if (!rdo::File::exist(fileName))
	{
		::MessageBox(NULL, rdo::format(ID_MSG_NO_HELP_FILE,
fileName.c_str()).c_str(), NULL, MB_ICONEXCLAMATION | MB_OK);
		return _T("");
	}
	return fileName;
}

QProcess* RDOStudioApp::chkQtAssistantWindow()
{
	if (!m_pAssistant)
	{
		return m_pAssistant = runQtAssistantWindow();
	}
	else if (m_pAssistant->state() == m_pAssistant->Running)
		return m_pAssistant;
	else
		return m_pAssistant = runQtAssistantWindow();
}

QProcess* RDOStudioApp::runQtAssistantWindow() const
{
		QProcess *process = new QProcess;
		QStringList args;
		args << QLatin1String("-collectionFile")
			<< QLatin1String(getFullHelpFileName().c_str())
			<< QLatin1String("-enableRemoteControl")
			<< QLatin1String("-quiet");
		process->start(QLatin1String("assistant"), args);
		if (process->state() == process->Running)
			return process;}
```