Р О С С И Й С К А Я Ф Е Д Е Р А Ц И Я

М о с к о в с к а я о б л а с т ь

Филиал «Котельники»

Государственного бюджетного образовательного учреждения высшего образования Московской области

«Международный университет природы, общества и человека «Дубна»

Кафедра «Информационных технологий в управлении»

Курсовая работа

по дисциплине «Программирование на языке высокого уровня»

На тему "Разработка Web-браузера"

Студент: Юнькин Петр Дмитриевич.

Группа: ИВТ-11

Преподаватель: Сахаров Михаил Викторович

Котельники 2015

Оглавление

Введение 3
Глава 1. Анализ предметной области 4
1.1 Требования к браузеру 4
1.2 История развития браузеров 4
1.3 Текущее положение на рынке браузеров 5
Вывод по Главе 1 7
Глава 2. Выбор программных средств для реализации 7
2.1 Выбор браузерного движка 7
2.2 Выбор языка программирования и среды для разработки. Библиотека QT и среда разработки QtCreator. 8
Выводы по Главе 2 13
Глава 3. Реализация программы 13
3.1 Определение класса MainWindow 13
3.2 Конструктор класса MainWindow 15
3.3 Реализация методов класса 18
Выводы по Главе 3: 21
Глава 4. Описание пользовательского интерфейса и функционала 21
Выводы по Главе 4 24
Заключение 25
Список литературы 25

Введение

Всеобщая информатизация нашего общества в последние годы привела к тому, что сейчас мы получаем релевантную информацию преимущественно из Интернета.

Для обеспечения комфортного получения информации требуется наличие соответствующего программного обеспечения – браузера.

История развития браузеров началась в 1990 г. с создания первого браузера WorldWideWeb. Но прогресс не стоит на месте. Сегодня браузер – это не только средство для получения информации, но и программа, способная предоставить функционал для работы, общения и развлечения. В браузере можно использовать различные текстовые и графические редакторы, мессенджеры и чаты, среды для программирования, есть возможность потокового воспроизведения аудио и видео файлов, а также играть в простые игры.

Целью данной курсовой работы является разработка браузера, имеющего базовый функционал для получения информации из сети Интернет и понятный для пользователя интерфейс.

Для реализации программы была использована библиотека Qt5, основанная на C++.

Глава 1. Анализ предметной области

1.1 Требования к браузеру

Браузер - это программа, предназначенная для просмотра Web-страниц, управления Web-приложениями и решения других задач.

Основными требованиями к браузеру являются:

1. Корректное отображение Web-сайтов без потери верстки и заданного функционала при использовании различных устройств;
2. Обеспечение базовых средств навигации между Web-страницами;
3. Стабильная работа программы на любых сайтах и при любых сценариях использования;
4. Предоставление средств для отображения мультимедийных и иных объектов, входящих в состав Web-страниц.

1.2 История развития браузеров

Первый Web-браузером стал WorldWideWeb (Nexus), разработанный Тимом Бернерс-Ли. Но первым из браузеров с графическим интерфейсом распространение получил NCSA Mosaic, код которого был открыт и в дальнейшем был использован для создания таких браузеров как Netscape Navigator и Internet Explorer. Netscape, в том числе и с помощью выходцев из NCSA выпустила Netscape Navigator для разных операционных систем, пользовавшимся коммерческим успехом. Это побудили Microsoft к написанию собственного браузера Internet Explorer.

В 1995 году Microsoft выпустила ОС Windows 95, получившую при обновлении встроенный браузер Internet Explorer 3.0. При этом Microsoft

добавила в свой браузер несовместимые со стандартами расширения языка HTML, что стало началом «войны браузеров». В дальнейшем прозошла монополизация (около 95%) рынка браузером Microsoft.

После монополизации рынка Microsoft почти прекратила развитие своего браузера, со временем стал отставать по качеству от конкурентов, таких как Mozilla Foundation (браузер Firefox) и Opera Software (Opera Browser).

В 2008 году компания Google решила «помочь» конкуренции на рынке браузеров и выпустила свой браузер — Chrome, основанный на свободном проекте Chromium, ставшим очень популярным. Большинство современных браузеров основано именно на этом проекте.

1.3 Текущее положение на рынке браузеров

На текущий момент на рынке браузеров существует множество решений, разрабатываемых разными компаниями.

Основными игроками рынка браузеров для десктопов являются:

- Google Inc. (Google Chrome)
- Mozilla Foundation (Mozilla Firefox)
- Microsoft Corporation (Internet Explorer)
- Opera Software (Opera Browser)
- Apple Inc. (Safari)

Также в последнее время сильно увеличилась доля использования мобильных браузеров. Из-за роста количества и качества мобильных устройств и повсеместного внедрения высокоскоростных мобильных сетей пользователи все больше используют свои личные мобильные устройства для поиска информации в Сети. Еще одним драйвером роста использования

Интернета с мобильных устройств являются пользователи из развивающихся стран Азии и Африки, не имеющие средств на покупку полноценного ПК.

Основными браузерами для мобильных устройств являются:

1. Google Chrome
2. Базовый браузер ОС Android
3. Базовый браузер ОС iOS
4. Opera Mobile/Mini

Ниже на Рис. 1 и Рис.2 представлена инфографика, показывающая доли использования современных браузеров (по подсчетам StatCounter):

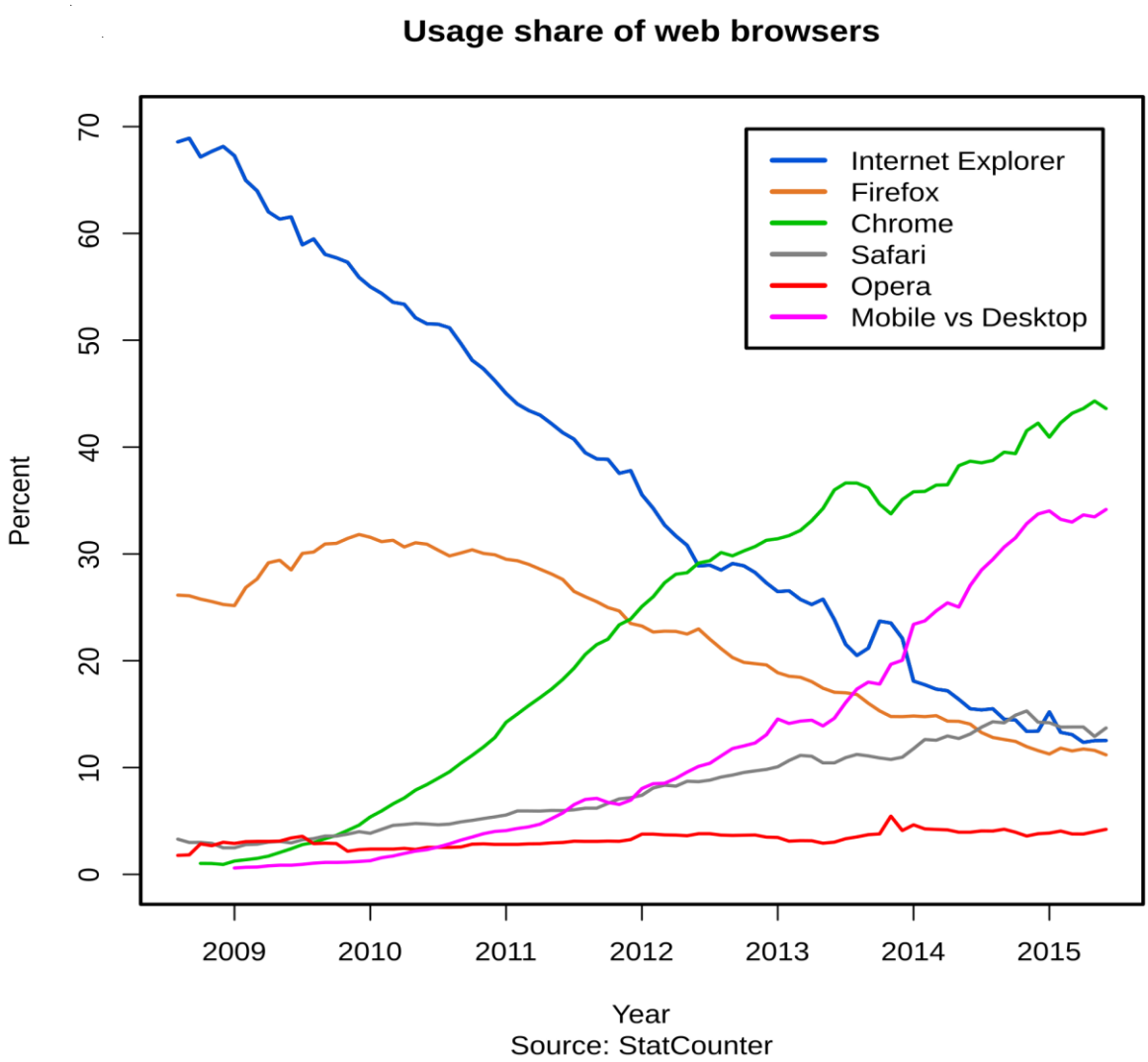

Рисунок 1. Долевое распределение десктопных браузеров

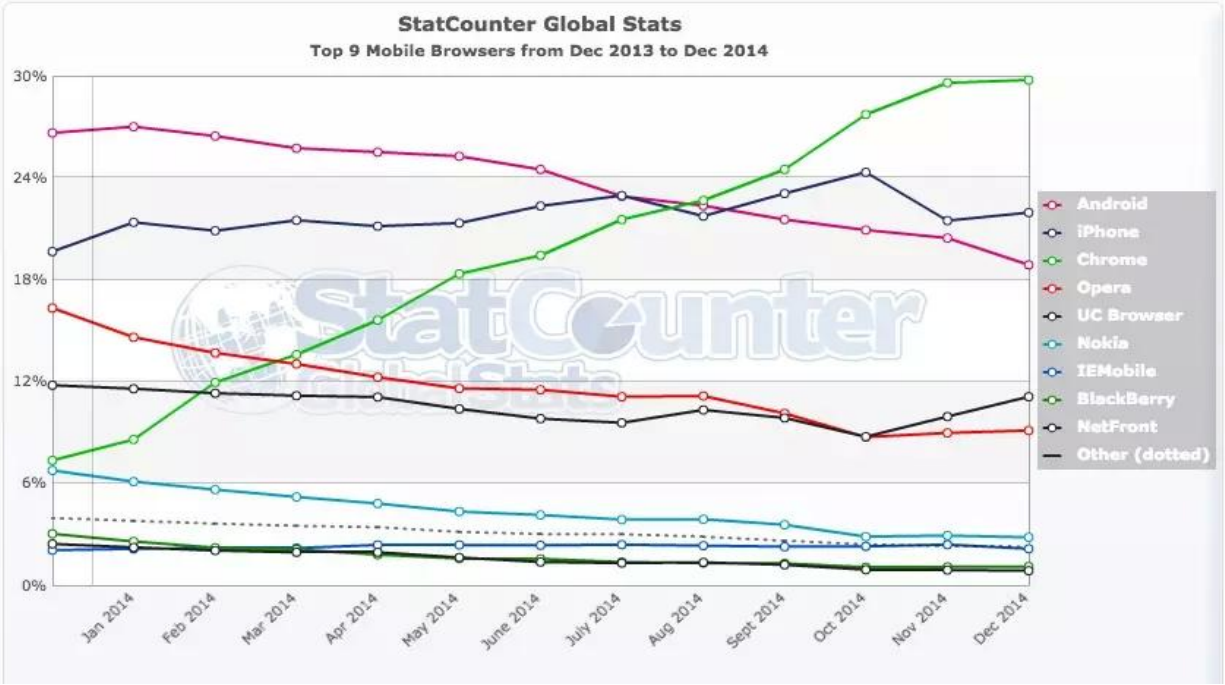

Рисунок 2. Долевое распределение мобильных браузеров

Вывод по Главе 1

В данной главе мы ознакомились с понятием браузера и основными требованиями, предъявляемыми к нему. Также была изучена история создания браузера и текущие тенденции на рынке браузеров.

Глава 2. Выбор программных средств для реализации

2.1 Выбор браузерного движка

Для обеспечения основной функции браузера – просмотра Web-страниц требуется браузерный движок.

Браузерный движок (англ. layout engine) — представляет собой программу, преобразующую содержимое Web-страниц (файлы HTML, XML, цифровые изображения и т. д.) и информацию о форматировании (в форматах CSS, XSL и т. д.) в интерактивное изображение форматированного содержимого на экране.

Написание собственного браузерного движка является достаточно трудоёмкой задачей даже для крупных IT компаний. Поэтому одной из задач,

решаемых в данной работе, является выбор подходящего браузерного движка из свободно доступных на данный момент.

На данный момент основными движками для браузера являются:

Trident — проприетарный движок Microsoft Internet Explorer; используется многими программами для Microsoft Windows (например, мини-браузерами в программах Winamp и RealPlayer).

Gecko — открытый движок проекта Mozilla; используется в большом числе программ, основанных на коде Mozilla (браузере Firefox, почтовом клиенте Thunderbird, наборе программ SeaMonkey).

WebKit — движок для браузера Apple Safari, включенного в операционную систему Mac OS X, и браузера Google Chrome.

Blink - движок браузера Google Chrome с 28 версии и Opera 15. Является переработанным WebKit.

Для решения задачи был выбран движок WebKit. WebKit на данный момент поддерживает огромное количество актуальных Web-стандартов, требующихся для корректного отображения Web-страниц и работы Web-приложений на HTML5. При этом WebKit является свободным ПО, что позволяет сообществу разработчиков улучшать его и бесплатно использовать в своих проектах.

2.2 Выбор языка программирования и среды для разработки. Библиотека QT и среда разработки QtCreator.

Для написания программы была выбрана библиотека Qt5 – кроссплатформенный инструментарий для разработки ПО на языке C++ и среда для разработки «Qt Creator».

Со времени своего появления в 1996 году библиотека Qt легла в основу тысяч успешных проектов во всём мире. Qt используется в Autodesk Maya, Adobe Photoshop Elements, OPIE, Skype, Медиапроигрыватель VLC, VirtualBox, Mathematica и других программах.

Qt 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.

Отличительная особенность Qt от других библиотек — использование Meta Object Compiler (MOC) — предварительной системы обработки исходного кода (в общем-то, Qt — это библиотека не для чистого C++, а для его особого наречия, с которого и «переводит» MOC для последующей компиляции любым стандартным C++ компилятором). MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на C++ описания классов, содержащие макрос Q_OBJECT, и создаёт дополнительный исходный файл на C++, содержащий метаобъектный код. Сигналы и слоты используются для коммуникации между объектами. Механизм сигналов и слотов главная особенность Qt и вероятно та часть, которая отличается от особенностей, предоставляемых другими библиотеками.

При разработке графического интерфейса, меняя виджет, мы часто хотим что бы другой виджет получил об этом уведомление. В общем, мы хотим того, чтобы любые объекты любых типов могли общаться с другими. Например, при нажатии кнопки «Закрыть», мы скорее всего ожидаем вызова функции окна close().

Другие библиотеки реализуют общение между объектами с помощью обратного вызова. Обратный вызов является указателем на функцию, то есть, если требуется что бы функция уведомила нас о каких-нибудь событиях, мы передаем указатель на другую функцию (обратновызываемую) этой функции. Функция в таком случае делает обратный вызов когда необходимо. Обратный вызов имеет два основных недостатка. Во-первых, он не является типобезопасным. Мы никогда не можем быть уверены что функция делает обратный вызов с корректными аргументами. Во-вторых, обратный вызов жестко связан с вызывающей его функцией, так как эта функция должна точно знать какой обратный вызов надо делать.

В Qt применяется другой механизм — сигналы и слоты. Сигнал вырабатывается когда происходит определенное событие. Слот является функцией, которая вызывается в ответ на определенный сигнал. Виджеты Qt имеют много предопределенных сигналов и слотов, но мы всегда можем сделать дочерний класс и добавить наши сигналы и слоты в нем.

Механизм сигналов и слотов типобезопасен. Сигнатура сигнала должна совпадать с сигнатурой слота-получателя. (Фактически слот может иметь более короткую сигнатуру чем сигнал который он получает, так как он может игнорировать дополнительные аргументы). Так как сигнатуры сравнимы, компилятор может помочь нам обнаружить несовпадение типов. Сигналы и слоты слабо связаны. Класс, который вырабатывает сигнал не знает и не

заботится о том, какие слоты его получат. Механизм сигналов и слотов Qt гарантирует, что если мы подключим сигнал к слоту, слот будет вызван с параметрами сигнала в нужное время. Сигналы и слоты могут принимать любое число аргументов любого типа. Они полностью типобезопасны.

Все классы, наследуемые от QObject или его дочерних классов (например, QWidget) могут содержать сигналы и слоты. Сигналы вырабатываются объектами когда они изменяют свое состояние так, что это может заинтересовать другие объекты. При этом он на знает и не заботится о том что у его сигнала может не быть получателя.

Слоты могут быть использованы для получения сигналов, но они так же нормальные функции-члены. Так же как объект не знает ничего о получателях своих сигналов, слот ничего не знает о сигналах, которые к нему подключены. Это гарантирует что полностью независимые компоненты могут быть созданы с помощью Qt.

Мы можем подключать к одному слоту столько сигналов, сколько захотим, также один сигнал может быть подключен к стольким слотам, сколько необходимо. Так же возможно подключать сигнал к другому сигналу (это вызовет выработку второго сигнала немедленно после появления первого).

Сигналы и слоты вместе составляют мощный механизм создания компонентов.

Одним из весомых преимуществ проекта Qt является наличие качественной документации. Статьи документации снабжены большим количеством примеров. Исходный код самой библиотеки хорошо форматирован, подробно комментирован и легко читается, что также упрощает изучение Qt.

В состав Qt5 входят модули для разработки Web-приложений:

- QtWebKitWidgets (движок WebKit)
- QtWebEngine (с версии 5.4) (движок Blink)

Для написания программы был использован модуль QtWebKitWidgets, т.к. при испозовании модуля QtWebEngine могут возникнуть сложности из-за того, что данный модуль был введен недавно и еще недостаточно стабилен.

Среда разработки «Qt Creator» (скриншот представлен на Рис.3), которая включает в себя редактор кода, справку, графические средства «Qt Designer» и возможность отладки приложений. «Qt Creator» может использовать GCC или Microsoft VC++ в качестве компилятора и GDB в качестве отладчика. Для Windows версий библиотека комплектуется компилятором, заголовочными и объектными файлами MinGW.

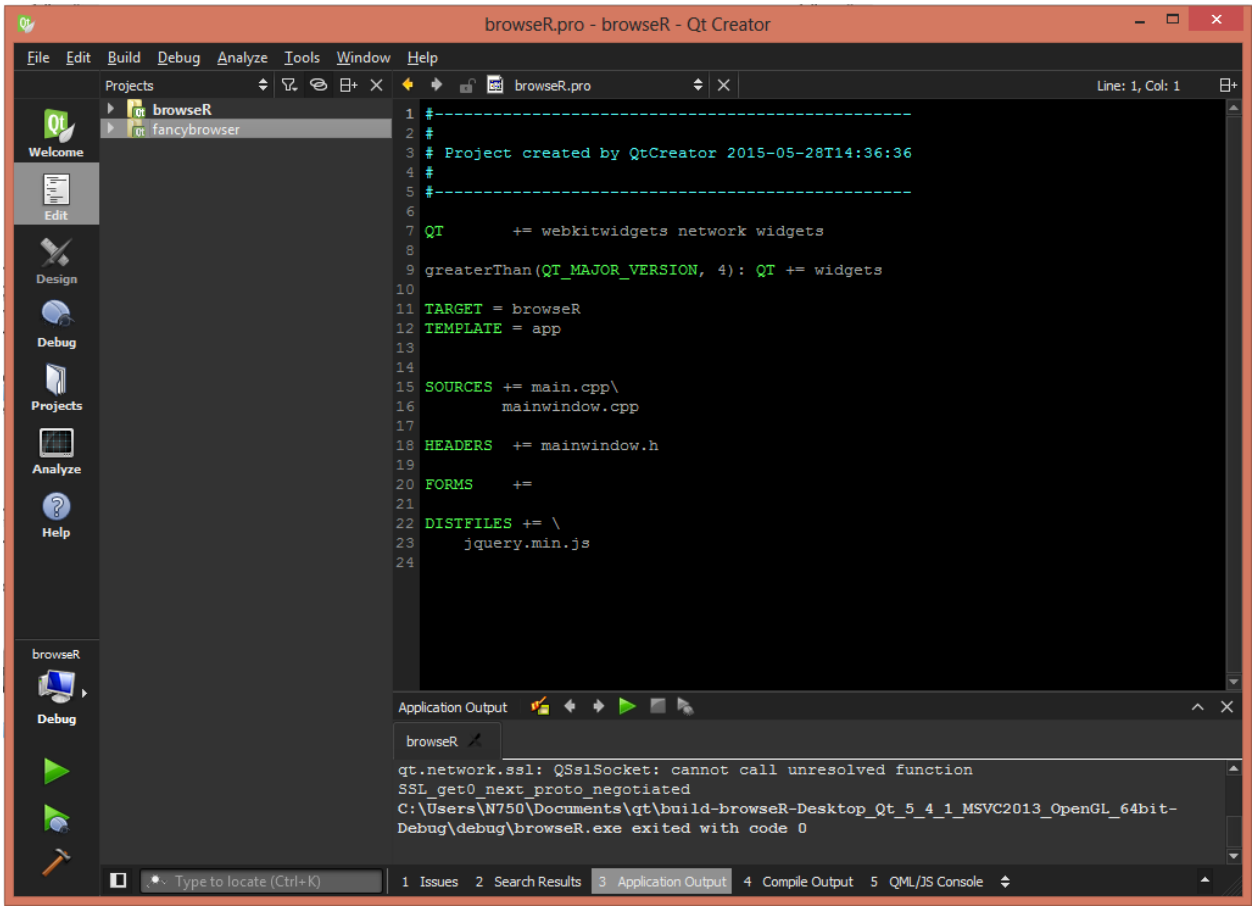

Рисунок 3. Среда разработки QtCreator

Выводы по Главе 2

В данной главе были выбраны и рассмотрены программные средства для разработки Web-браузера – библиотека Qt и среда разработки QtCreator. Приведена аргументация в пользу выбора данных средств. Учтены особенности инструментария для более простого выполнения поставленной задачи.

Глава 3. Реализация программы

3.1 Определение класса MainWindow

Опеределение класса MainWindow происходит в заголовочном файле.

Программа использует QWebFrame::evaluateJavaScript для встраивания jQuery JavaScript кода. QMainWindow с QWebView в качестве центрального виджета производит построение браузера сам.

Класс MainWindow наследуется от класса QMainWindow. Это позволяет слотам выполнять действия и с приложением, и с Web-контентом.

```
class MainWindow : public QMainWindow

{

   Q_OBJECT


public:

   MainWindow(const QUrl& url);


protected slots:
```

```
    void adjustLocation();
    void changeLocation();
    void book1show();
    void book2show();
    void book3show();
    void adjustTitle();
    void setProgress(int p);
    void finishLoading(bool);
    void viewSource();
    void slotSourceDownloaded();
    void highlightAllLinks();
    void rotateImages(bool invert);

private:
    QString jQuery;
    QWebView *view;
    QLineEdit *locationEdit;
    QAction *rotateAction;
    int progress;
};
```

Также объявляем QString, который содержит jQuery, QWebView, показывающий Web-контент и QLineEdit, являющийся адресной строкой.

3.2 Конструктор класса MainWindow

```
MainWindow::MainWindow(const QUrl& url)
{
    progress = 0;

    QFile file;
    file.setFileName(":/jquery.min.js");
    file.open(QIODevice::ReadOnly);
    jQuery = file.readAll();
    jQuery.append("\nvar qt = { 'jQuery': jQuery.noConflict(true) };");
    file.close();
```

В первой части конструктора значение переменной progress приравнивается к нулю. Это значение будет использовано в коде позже для визуализации загрузки Web-страницы.

Далее загружаем файл библиотеки jQuery, используя QFile и считываем его.

Библиотека jQuery – JavaScript библиотека, которая обеспечивает функции для манипулирования HTML.

```
view = new QWebView(this);
view->load(url);
connect(view, SIGNAL(loadFinished(bool)), SLOT(adjustLocation()));
connect(view, SIGNAL(titleChanged(QString)), SLOT(adjustTitle()));
```

```
connect(view, SIGNAL(loadProgress(int)), SLOT(setProgress(int)));
connect(view, SIGNAL(loadFinished(bool)), SLOT(finishLoading(bool)));
locationEdit = new QLineEdit(this);
locationEdit->setSizePolicy(QSizePolicy::Expanding,
locationEdit->sizePolicy().verticalPolicy());
connect(locationEdit, SIGNAL(returnPressed()), SLOT(changeLocation()));

QToolBar *toolBar = addToolBar(tr("Navigation"));
toolBar->addAction(view->pageAction(QWebPage::Back));
toolBar->addAction(view->pageAction(QWebPage::Forward));
toolBar->addAction(view->pageAction(QWebPage::Reload));
toolBar->addAction(view->pageAction(QWebPage::Stop));
toolBar->addWidget(locationEdit);
```

Вторая часть конструктора создает QWebView и соединяет слоты для просмотра сигналов. Кроме того, мы создаем QLineEdit в качестве адресной строки. После мы задаем горизонтальную QSizePolicy чтобы свободное пространство в браузере было заполнено полностью. Добавляем QLineEdit в QToolBar вместе с набором навигационных действий из QWebView::pageAction.

```
QMenu *viewMenu = menuBar()->addMenu(tr("&View"));
QAction* viewSourceAction = new QAction("Page Source", this);
connect(viewSourceAction, SIGNAL(triggered()), SLOT(viewSource()));
```

```
    viewMenu->addAction(viewSourceAction);
    QMenu *bookMenu = menuBar()->addMenu(tr("&Bookmarks"));
    bookMenu->addAction(tr("Google"), this, SLOT(book1show()));
    bookMenu->addAction(tr("Weather"), this, SLOT(book2show()));
    bookMenu->addAction(tr("YouTube"), this, SLOT(book3show()));

    QMenu *effectMenu = menuBar()->addMenu(tr("&Effect"));
    effectMenu->addAction("Highlight all links", this, SLOT(highlightAllLinks()));
    rotateAction = new QAction(this);
    rotateAction->setIcon
    (style()->standardIcon(QStyle::SP_FileDialogDetailedView));
    rotateAction->setCheckable(true);
    rotateAction->setText(tr("Turn images upside down"));
    connect(rotateAction, SIGNAL(toggled(bool)),
    this, SLOT(rotateImages(bool)));
    effectMenu->addAction(rotateAction);
    setCentralWidget(view);
}
```

Третья и последняя часть конструктора релизует три QMenu и назначает им набор действий. Последняя строка делает QWebView центральным виджетом в QMainWindow.

3.3 Реализация методов класса

```
void MainWindow::adjustLocation()
{
   locationEdit->setText(view->url().toString());
}


void MainWindow::changeLocation()
{
   QUrl url = QUrl::fromUserInput(locationEdit->text());
   view->load(url);
   view->setFocus();
}
```

Когда страница загружена adjustLocation() обновляет адресную строку,

adjustLocation() срабатывает по сигналу loadFinished() в QWebView. В changeLocation() мы создаем QUrl-объект и используем его для загрузки страницы в QwebView. Когда новая Web-страница завершила загрузку adjustLocation() будет запущен еще раз для обновления адресной строки.

Закладки book1show(), book2show(), book3show() реализованы аналогично changeLocation().

```
void MainWindow::adjustTitle()
{
   if (progress <= 0 || progress >= 100)
```

```
        setWindowTitle(view->title());
    else
        setWindowTitle(QString("%1 (%2%)").arg(view->title()).arg(progress));
}
void MainWindow::setProgress(int p)
{
    progress = p;
    adjustTitle();
}
```

adjustTitle() задает заголовок окна и показывает прогресс загрузки. Этот слот вызывается сигналом titleChanged() в QWebView.

```
void MainWindow::finishLoading(bool)
{
    progress = 100;
    adjustTitle();
    view->page()->mainFrame()->evaluateJavaScript(jQuery);
    rotateImages(rotateAction->isChecked());
}
```

Когда Web-страница загружена finishLoading() вызывается сигналом loadFinished() в QWebView. Далее finishLoading() обновляет прогресс в

заголовке окна и вызывает evaluateJavaScript(jQuery), что позволяет исполнять JavaScript код. Код JavaScript будет представлен как часть контента, загруженного в QWebView, и после должен быть загружен каждый раз при загрузке новой страницы. Если библиотека однажды загружена, мы можем начать выполнение разных jQuery функций в браузере.

Функция rotateImages() вызывается чтобы удостовериться, что изображения на новой загруженной странице соответствовали состоянию действия переворота.

```
void MainWindow::highlightAllLinks()
{
    QString code = "qt.jQuery('a').each( function ()
 {
 qt.jQuery(this).css('background-color', 'yellow') } ); undefined";
    view->page()->mainFrame()->evaluateJavaScript(code);
}
```

Первая функция, основанная на jQuery, выделяет все ссылки на текущей Web-странице. JavaScript код ищет Web-элемент с названием a, который является меткой гиперссылки. Каждой гиперссылке присваивается желтый цвет фона с помощью CSS.

```
void MainWindow::rotateImages(bool invert)
{
    QString code;
    if (invert)
```

```
    code = "qt.jQuery('img').each( function () { qt.jQuery(this).css('-webkit-transition', '-webkit-transform 2s'); qt.jQuery(this).css('-webkit-transform', 'rotate(180deg)') } ); undefined";
  else
    code = "qt.jQuery('img').each( function () { qt.jQuery(this).css('-webkit-transition', '-webkit-transform 2s'); qt.jQuery(this).css('-webkit-transform', 'rotate(0deg)') } ); undefined";
  view->page()->mainFrame()->evaluateJavaScript(code);
}
```

Функция rotateImages() переворачивает изображения на текущей Web-странице. WebKit поддерживает преобразования CSS и этот JavaScript код ищет все img элементы и поворачивает все изображения на 180 градусов и обратно.

Выводы по Главе 3:

В данной главе были рассмотрены и разобраны самые важные участки кода программы, раскрывающие возможности выбранного программного средства. Был рассмотрен класс главного окна MainWindow, его конструктор и функции, в которых реализованы основной функционал.

Глава 4. Описание пользовательского интерфейса и функционала

Пользовательский интерфейс браузера (показан на Рис.4) достаточно прост и не вызовет затруднения при использовании у рядового пользователя.

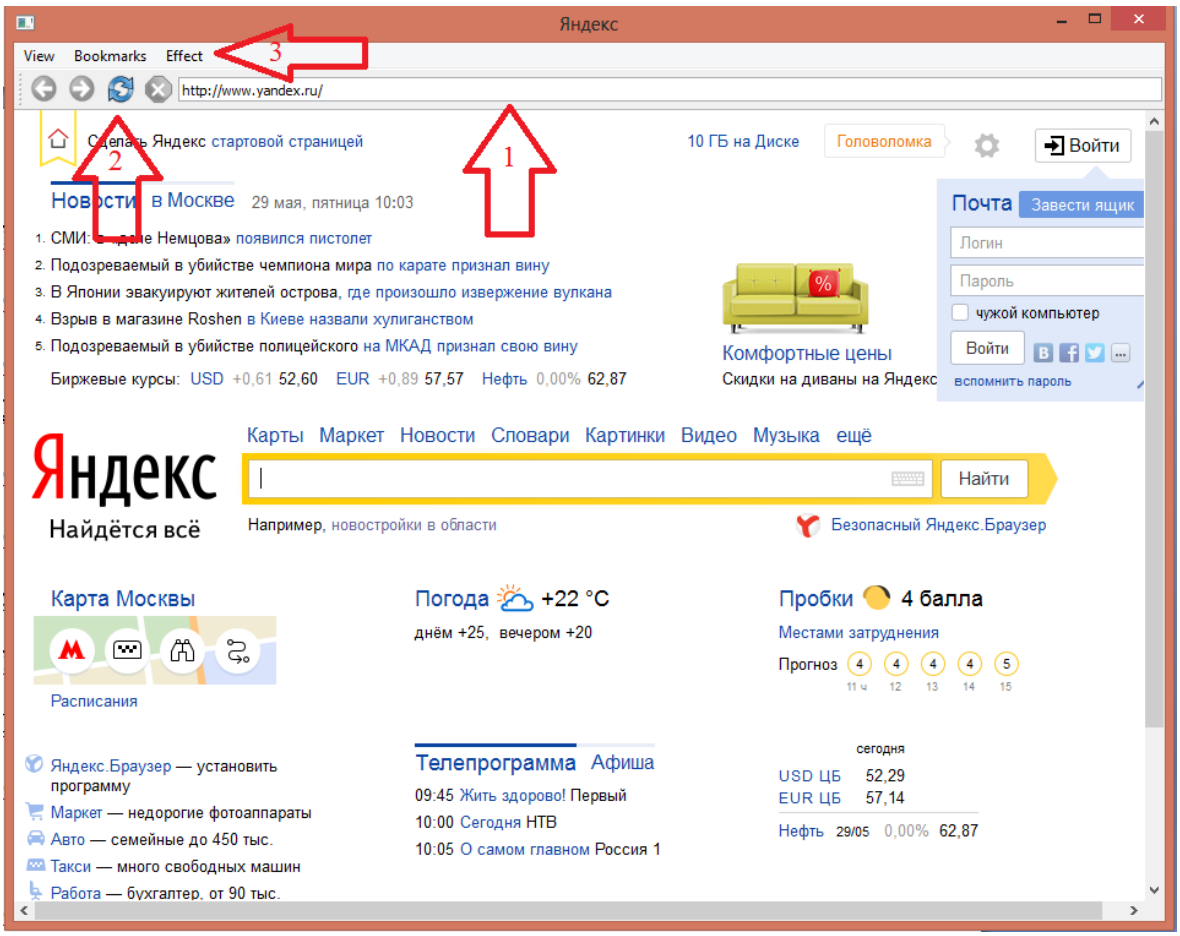

Рисунок 4. Основной интерфейс браузера

В программе был реализован следующий функционал:

- Адресная строка браузера (1)
- Тулбар со средствами навигации по страницам (2)
- Меню с выпадающим списком, содержащее разные функции (3)

В адресную строку вводится Web-адрес сайта, на который нужно перейти.

Средства навигации в тулбаре выполняют следующие операции:

- Переход на предыдущую страницу (кнопка «Go Back»)
- Переход на следующую страницу (кнопка «Go Forward»)
- Обновление страницы (кнопка Reload)
- Прерывание загрузки текущей страницы (кнопка Stop)

Каждая из кнопок снабжена соответствующей иконкой.

Меню состоит из трех пунктов:

1. View
2. Bookmarks
3. Effect

В первом пункте меню при нажатии появляется пункт Page Source, при нажатии на который появляется окно с исходным кодом Web-страницы (Рис.5).

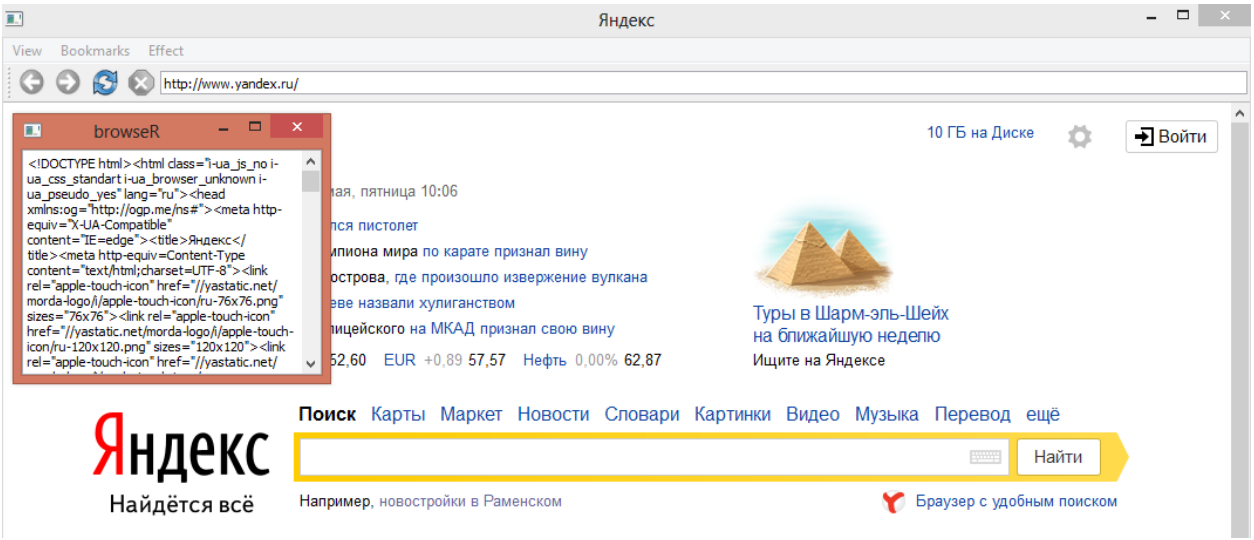

Рисунок 5. Окно с исходным кодом страницы

Во втором пункте реализован список предустановленных закладок для перехода на часто используемые сайты.

Последний пункт содержит в себе функции, использующие библиотеку jQuery и JavaScript код.

Первая функция выделяет все ссылки на странице, а вторая – переворачивает изображения на 180 градусов.

Данные функции представлены для демонстрации возможности дальнейшей доработки программы, используя вставки кода на языке JavaScript. Работа этих функции представлена ниже на Рис.6 и Рис.7.

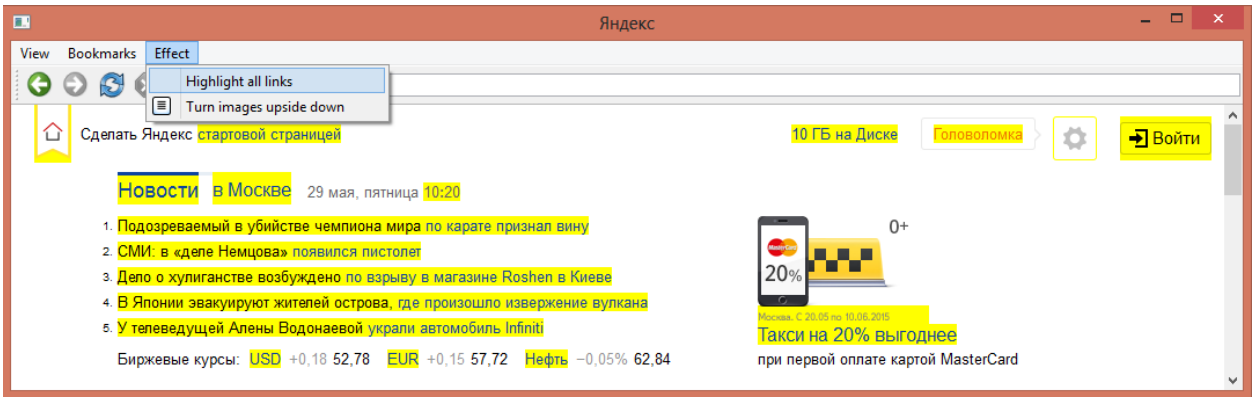

Рисунок 6. Выделение ссылок

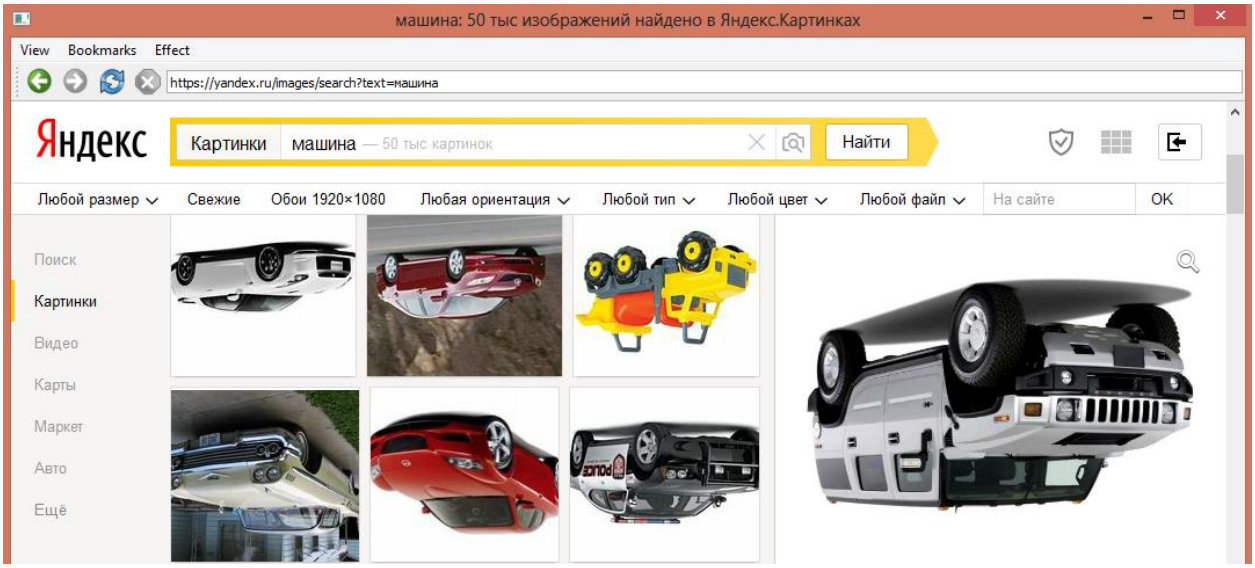

Рисунок 7. Переворот изображений

Выводы по Главе 4

В данной главе был рассмотрен интерфейс браузера и основной функционал.

Заключение

В результате разработки программного средства, была написана программа для просмотра Web-страниц и их содержимого. Программа имеет простой понятный для пользователя интерфейс, обеспечивает беспрепятственную навигацию по Web-страницам и может использовать как рабочий инструмент или мультимедийный проигрыватель благодаря поддержке Web-приложений на HTML5.

Программа разрабатывалась с использованием библиотеки Qt, что позволило обойтись без системного кода, созданного средой разработки, который усложняет визуальное восприятие программы. При этом Qt является кроссплатформенной библиотекой и позволяет компилировать один и тот же код на разных платформах и операционных системах. Это Qt позволяет писать один код программы для нескольких устройств одновременно.

Список литературы

1. Интернет-энциклопедия Wikipedia ru.wikipedia.org
2. Макс Шлее «Qt 5.3. Профессиональное программирование на C++»
3. Макс Шлее «Qt 4.8. Профессиональное программирование на C++»
4. Жасмин Бланшет, Марк Саммерфилд «Qt 4: программирование GUI на C++»
5. Портал «Хабрахабр» www.habrahabr.ru