

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. ОБЩИЙ РАЗДЕЛ.....	5
1.1. Системные требования.....	5
1.2. Характеристика системы программирования.....	6
1.3. Требования к программному продукту.....	8
2. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ.....	8
2.1. Постановка задачи	9
2.2. Информационная модель программы решения задачи	10
2.3. Логическая модель программы	12
2.4. Тестирование программы.....	24
2.5. Анализ результатов тестирования.....	26
3. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ.....	27
3.1. Руководство программиста.....	27
3.2. Руководство пользователя.....	31
ЗАКЛЮЧЕНИЕ.....	37
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	38
ПРИЛОЖЕНИЕ А.....	39

ВВЕДЕНИЕ

На текущий момент на рынке программного обеспечения нет универсального кроссплатформенного архиватора, который имел бы одинаковый графический пользовательский интерфейс на разных платформах и позволял пользователю без изучения новой программы создать архив с нужными ему файлами. Кроме этого, на некоторых операционных системах в предустановленных программах нет архиватора с графическим пользовательским интерфейсом вовсе (Gentoo linux). Целью данной работы является получение новых знаний и результатов в области работы с объектно-ориентированным программированием, свойств различных операционных систем, исследование потребностей пользователей в программном обеспечении и построении интуитивно понятного графического пользовательского интерфейса, создание программы “Архиватор”, которая решит проблему. На момент 29 октября 2017 года в подобный архиватор не создан. Действительно, многим пользователям жизненно необходим подобный архиватор. Архиватор должен решать несколько задач, а именно: создания своего диалогового окна выбора файлов, не зависящего от различных операционных систем, создание универсального пользовательского интерфейса, меняющего свой дизайн под стиль наивных компонентов операционной системы. Объектами исследования и разработки в данной работе являются библиотеки для кроссплатформенного программирования, написанные на языке программирования C++, а так же особенности операционных систем. Работа будет выполнена при использовании современных технологий объектно-ориентированной разработки программного обеспечения в сочетании с использованием готовых классов-прослоек для нативной работы на разных операционных системах. Будет использована библиотека Qt5, технологии метакомпиляции, последние стандарты C++11. Будет произведена работа по созданию универсального архиватора, с возможностью компиляции его исходного кода под разные операционные

системы с единым пользовательским интерфейсом, упрощающим работу с архиватором на разных платформах. Это будет достигнуто благодаря использованию библиотеки Qt. В перспективе планируется добавление архиватора в качестве предустановленного пакета в отечественные операционные системы. В качестве формата сжатия файла будет выбран алгоритм архивации файлов Zip, как наиболее популярный среди пользователей электронно вычислительных систем в Российской Федерации. Данный формат позволяет сжимать файлы “Без потерь”, что позволяет использовать его в большинстве задач, решаемых рядовым пользователем компьютера. Формат был создан в 1989 году и проверен временем. Кроме этого, существуют сторонние библиотеки для языка программирования C++, которые позволяют сжимать файлы и отвечают требованиям нашего решения. Для сжатия данных в программе будет использована библиотека с открытым исходным кодом zlib, так как она интегрирована в Qt и является кроссплатформенной. Лицензия этой библиотеки позволяет использовать её в стороннем программном обеспечении, сообщество открытого исходного кода считает её свободной. На текущий момент эту библиотеку используют крупные корпорации при построении своих сайтов для сжатия картинок в формате png при передачи данных от сервера к клиенту. Библиотека проявила себя как легковесное универсальное средство для сжатия файлов, которая подходит для решения простых задач широкой специализации, например, для сжатия музыки лучше использовать другой алгоритм кодирования звуковой волны, но эта библиотека позволяет сжать любой файл, не зависимо от его структуры, так как обрабатывает его исключительно на уровне массива битов. Подобный вариант решения задачи сжатия идеально вписывается в задумку программы, так как средство сможет сжать любой файл, не зависимо от даты создания его формата и сделает это на любой операционной системе, где возможно её скомпилировать и запустить. Это добавит решает проблему универсальности и сделает средство необходимым инструментом в работе рядового пользователя компьютера.

ОБЩИЙ РАЗДЕЛ

1.1 СИСТЕМНЫЕ ТРЕБОВАНИЯ

В процессе создания программы был использован компьютер с предустановленной операционной системой Windows 10 сборки 15063.674 редакции “Домашняя” с 8 гигабайтами ОЗУ, 64 разрядным процессором Intel Core I5-4460 и установленной средой Qt Creator 4.4.1 с компилятором MinGW32 и версией библиотеки Qt 5.9. Эти требования можно назвать рекомендуемыми. Свободное место на диске C было выше 120 гигабайт, имя пользователя было задано английскими буквами. Теоретически, исходный код можно собрать на любой операционной системе, которую поддерживает библиотека Qt и на которой есть возможность установить компилятор GCC, или его аналог как MinGW32, или другой компилятор, который поддерживает библиотека Qt, но работа в данном случае не может гарантироваться. Стоит добавить, что физические минимальные системные требования наследуются от использованных в процессе разработки сторонних библиотек и могут отличаться в зависимости от разных операционных систем. На операционной системе Windows, при использовании компилятора MinGW, все динамично подключаемые библиотеки должны лежать в директории с исполняемым файлом. Это позволяет запускать программы без доустановки дополнительных модулей, что понижает системные требования – для установки не нужны права администратора. Но если пересобрать программу с использованием компилятора MS Build из Visual C++, физические системные требования сократятся, но появится обязательное требование наличие права администратора – без установки дополнительного ПО Visual Studio Redistributable соответствующей году выпуска поставляемой с компилятором Visual Studio 32 разрядной при сборке под архитектуру x86 и 32,64 разрядной при сборке под архитектуру 64 исполняемый файл не сможет запуститься, так как операционная система не сможет подгрузить все необходимые библиотеки.

1.2 ХАРАКТЕРИСТИКА СИСТЕМЫ ПРОГРАММИРОВАНИЯ

В качестве языка программирования был выбран язык C++, так как он императивный: описывает не задачу, а способ ее решения; объектно-ориентированный: весь язык представлен в виде объектов и к нему применимы понятия полиморфизма, наследования и инкапсуляции; обобщенный: алгоритму в C++ можно передать ему любые данные. Кроме этого C++ обладает статической типизацией: в нем невозможно изменить тип в процессе выполнения. Так же, типизация в C++ явная, это требует указывать тип параметров и переменных. Кроме этого, в C++ возможно неявное приведение типов. Это может быть полезно в процессе разработки. Благодаря этому, C++ лидирует у своих конкурентов и обгоняет другие языки программирования в отрасли, связанной с решением поставленной задачи.

В качестве компилятора будет выбран MinGW32. Это компилятор с открытым исходным кодом, который предоставляет часть своих динамически подключаемых библиотек вместе с собранным бинарным файлом. Выбор этого компилятора позволит сделать архиватор независимым от пакета библиотек Microsoft Visual Studio Redistributable, что упростит запуск программы. Вместе с компилятором MinGW32 в комплекте идет отладчик программ GNU gdb. Он позволит выполнять код программы пошагово, проверить общую работоспособность программы.

Вместе с библиотекой Qt поставляется встроенное средство для отрисовки окон Qt Widgets. Qt Widgets используют для отрисовки элементов управления свободную библиотеку отрисовки графики OpenGL, а не нативные вызовы API операционной системы. Это добавляет зависимость программы от поддержки видеокарты библиотекой OpenGL. Дискретная графика процессора Intel Core i3 поддерживает эту библиотеку и обеспечивает стабильную работу.

Кроме этого, архиватору необходим полный доступ с правами на чтение и запись к файлу и месту, с которым будет происходить работа. Почти во всех операционных системах, доступ к файлам разграничен по правам. Это касается

и Windows и операционных систем на базе ядер Linux, Unix. Если программа не получит доступ к файлу, архивация будет невозможна. Библиотека Qt позволит понять, что требуются привилегированные права к файлу, программа сможет оповестить пользователя. Но архивация при этом не произойдет.

В библиотеке Qt использование юнит тестов без автогенерации кода невозможно, поэтому для отладки и проверки работоспособности будет использован вывод в консоль. Причина этого обязательное добавление посторонних служебных слов в исходный код программы, которые не являются частью стандарта C++11. При создании сборки Release эта отладочный вывод не будет включен в ассемблерный код программы. Это будет достигнуто через использование директив препроцессора.

Для того, чтобы использовать только служебные слова стандарта C++11 в исходном коде программы к сигналам графических компонентов Qt Widgets будут подсоединяться не слоты, а лямбда выражения с неявным параметром `this`. В процессе компиляции данные выражения будут собраны как отдельные функции, благодаря неявной передаче указателя на класс они смогут взаимодействовать с его публичными объектами и свойствами. Адрес на подобную функцию будет сразу передан в статичный метод `connect()` класса `QObject`, так как подобной функции не будет задан идентификатор, то её вызов из другого метода, для которого она не предназначена, будет невозможен.

В качестве интерфейса взаимодействия для программистов будет добавлена возможность выбора системным администратором своей базы данных. Эта особенность программы позволит настроить её под нужды конкретного предприятия, что в свою очередь увеличит множество сфер, где будет применимо это решение. Так же, у системного администратора появится выбор, вести историю архиваций локально на компьютере или глобально на специальном сервере. В руководстве для программиста будет подробно расписано, как подключить архиватор к альтернативной базе данных

1.3 Требования к программному продукту

Требования к функциональным характеристикам

Программа должна осуществлять бесперебойную работу при создании архива из выбранных файлов, размер каждого файла до 1024 байт и полный доступ на чтение и запись, извлечение файлов из архива данного размера, а также включать в файл служебную информацию о компьютере, где был создан архив.

Программа должна:

- Работать с заданным алгоритмом функционирования;
- Поддерживать диалоговый режим в рамках предоставляемых пользователю возможностей;
- Производить бесперебойную работу по преобразованию информации.

Требования к надежности

- Данное программное обеспечение должно обеспечить показатель надежности от 90%. Показатель надежности будет определен по результатам тестирования
- Руководство к данному программному обеспечению должно включать перечень аварийных ситуаций, которые может решить системный администратор и инструкцию по их решению
- 1.3.3. Условия эксплуатации и требования к составу и параметрам технических средств
- Условия эксплуатации программы совпадают с условиями эксплуатации по ЭВМ IBM PC. Программа должна быть рассчитана на непрофессионального пользователя.

Минимальные требования к электронной вычислительной машине

- Процессор семейства Intel Core i7
- Оперативная память: 8Gb поколения DDR3
- Клавиатура и мышь.

Рекомендуемые требования к Электронной вычислительной машине

- Процессор семейства Intel Core i9
- Оперативная память: 16Gb поколения DDR4
- Клавиатура и мышь.

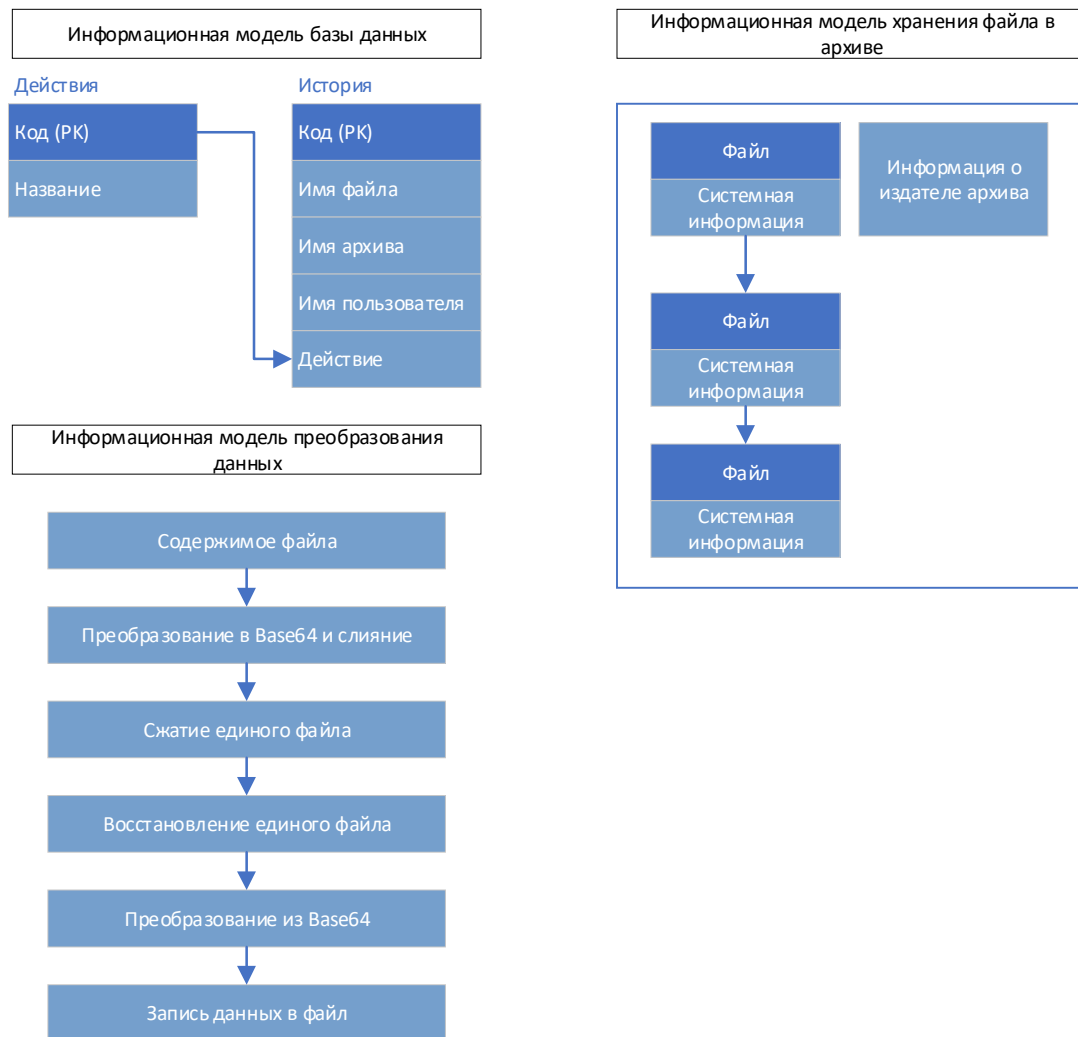
ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

2.1 ПОСТАНОВКА ЗАДАЧИ

При проведении практических работ в образовательных учреждениях существует проблема сдачи работ. В архиваторах, с которыми я имел опыт работать, нет возможности идентифицировать, на каком компьютере был сделан архив. Это позволяет учащимся сдать чужую работу как свою, просто переименовав архив, отправляемый преподавателю. Кроме этого, архив можно без ведения учёта разархивировать и изменить чужую работу. Это не дает преподавателю убедиться в том, что работа принадлежит непосредственно студенту, который её сдал. Исправить проблему можно создав новый архиватор со своим форматом архивации, который будет вести учёт создания и извлечения архивов. Для идентификации компьютера администратор должен будет задать уникальное имя для компьютера, указать драйвер и строку подключения для базы данных. Подобное решение будет очень гибко в настройке, можно будет вести или локальный аудит или глобальный, меняя настройки без перекомпиляции программы. Конечно, если появится неподдерживаемый драйвер базы данных, то его логику придется дописать к исходным кодам, но это решается путем перекомпиляции программы из исходных кодов, используя более современную версию библиотеки Qt. Кроме проверки на авторство, подобная система аналитики позволит узнать, какие файлы самые популярные, а затем оптимизировать потребление ресурсов компьютера, удалив программы, открывающие не используемые форматы файлов. Решить проблему пустой траты пространства на жестком диске не может ни один другой архиватор. Кроме этого, я не сталкивался с другими программами, не обязательно архиваторами, которые позволяют вести подобную аналитику. Кроме этого, свой формат архивации обеспечит невозможность открыть архив сторонними средствами, соответственно учет сжатия и извлечения файлов невозможно будет обойти, не имея исходного кода архиватора, отвечающего за сжатие.

2.2 ИНФОРМАЦИОННАЯ МОДЕЛЬ ПРОГРАММЫ РЕШЕНИЯ ЗАДАЧИ

Архиватор будет содержать две информационные модели хранения данных. Первая модель — это история архивации, которая будет записываться в базу данных. История архивации является отличительной особенностью этого архиватора, поскольку именно эта функция решает проблему, поставленную в предыдущем пункте. Поскольку строка подключения будет вынесена в файл настроек программы, а синтаксис SQL запросов для записи строк в таблицу одинаков на разных базах данных, подключить архиватор можно будет к любой базе данных. Системному программисту нужно будет сохранить строковые типы данных информационной модели строковыми в своем решении, таким образом у учреждения, где будет использоваться программа, появится выбор куда сохранять данные. Это может быть или локальная база данных Access или общая база данных для ведения глобального аудита MS SQL. Вторая модель — это специфичный для данного архиватора формат сжатия данных, который будет содержать список добавленных файлов не в виде древа каталогов, а в виде списка. Подобное решение не позволит студенту использовать альтернативные средства для получения содержимого архива, а значит обойти ведение истории архивации. Кроме этого, преподавателю будет удобно получать содержимое файлов. Так же, извлечь в одну директорию два архива с одного компьютера будет невозможно. Для достижения поставленной задачи в архив будет вшиваться псевдоним компьютера, извлечение файлов из архива будет происходить в автоматически создаваемую директорию, имя которой будет соответствовать псевдониму, а если папка уже есть, то будет выведена ошибка. Таким образом, индивидуальность каждого архива гарантирована. Студент не сможет переименовать чужой архив с работами и сдать как свою. А если он попытается извлечь и получить чужие работы, то его действия будут зафиксированы и преподаватель сможет узнать, где и когда чужой для студента архив был извлечен и использован с нечестными целями. Кроме этого, средство позволит вести учет наиболее часто используемых файлов.



Информационная модель программы.

2.3. Логическая модель программы

Для того, чтобы применить максимум технологий, было решено написать на Qt простой веб сервер и выводить пользовательский интерфейс на языке разметки html, отправляя его на сокет конечному пользователю как строку. Строка при этом генерируется в процессе работы программы на C++, а не в результате считывания файла, что позволит исключить проблему неопределенной кодировки консоли, сделать решение кроссплатформенным и исключить использование метакомпиляции, в отличие от создания форм через Qt Widgets, где в исходном коде класса нужно указывать специальные слова, удаляемые и заменяемые на генерируемый код в процессе сборки исходного кода компоновщиком. Мной был произведен анализ работы протокола HTTP. Я выяснил, что при открытии страницы веб браузер отправляет на сокет строку, где первое слово “GET”, второе обозначает запрашиваемую веб страницу после адреса хоста-сервера (например, http://127.0.0.1/index второе слово будет равно /index). Далее идет прочая информация, которая нас не интересует. После получения данных программа отправляет на сокет ответ, который состоит из “HTTP/1.0 200 Ok\r\n Content-Type: text/html; charset="utf-8"\r\n <html><head></head><body>...</body></html> "\r\n", который обязательно должен содержать в конце переход на новую строку. Ответ формируется в отдельном классе программы, где шаблоны страниц ответа формируются из константных полей типа данных “QString”. Подобный метод позволяет вшить страницы в саму программу и не даст злоумышленнику изменить их, что обеспечит стабильность работы программы.

Рассмотрим пример шаблона страницы информации. Код шаблона представляет собой константную строку QString, в которой в фигурных скобках помечены места для вставки текста (const QString infoPage = "<h2>{TITLE}</h2>
<p>{INFO}</p><hr>Назад");. В этой константной строке лежит код html, в содержимое тегов или атрибутов тегов вставлены “метки” в фигурных скобках, например {INFO}

В процессе формирования ответа программа динамично заменит подобные метки на текст, имеющий информационную значимость. Таким образом, код формирования ответа становится чище, так как он не включает в себя html код ответа, а лишь берет уже написанный код и подставляет туда данные. Кроме этого, подобное решение гораздо легче поддерживать стороннему программисту. Тем не менее, сделать шаблоны для всех страниц невозможно, так как в архиваторе есть динамически генерируемые таблицы html, в которые выводится содержимое текущего каталога с файлами. Зато, так как метку можно вставить и в атрибут тега, в шаблон можно передать часть логики. Это может быть гиперссылка или вставка кода на JavaScript, что будет интересно программистам-студентам, если те будут изучать мой проект как пособие. Так студенты смогут наглядно убедиться в том, что в одном проекте могут сочетаться несколько языков программирования. На этом формирование веб страниц можно считать разобранным.

Рассмотрим поиск нужной странице по получаемой ссылке. Выше я уже писал, что специальный класс программы получает содержимое ссылки после первого символа / (страница 10). Опытные веб программисты скажут, что ссылка – вещь динамичная, у ссылки могут быть GET параметры. Так, например ссылки `http://127.0.0.1:1488/fm/show?folder=C:\Users` и `http://127.0.0.1:1488/fm/show` должны указывать на одну и ту же страницу, просто в первом случае программа должна по ссылке получить значение переменной `folder` и вывести содержимое каталога `C:\Users`. Во втором же случае программа должна вывести содержимое стандартного каталога, который должен открываться при запуске программы, а не перехода из сторонней директории. Алгоритм получения подобных GET параметров я опишу ниже, сейчас разберем метод обработки ссылки.

В библиотеке Qt есть специальных класс `QStringList`. Он содержит массив элементов `QString` и несколько методов для их обработки, в том числе и добавление новых элементов. При этом создается новый массив, на 1 элемент

длиннее старого, в который перемещаются элементы старого массива и новый элемент, а затем старый массив уничтожается. Удаление элементов происходит по аналогии. Путем изучения технической документации, я нашел, у этой примитивной реализации коллекции через массив, метод `indexOf`. `indexOf` выводит номер элемента этой коллекции, который содержит в себе часть строки, передаваемой через параметр. Если подобной строки не существует, `indexOf` возвращает `-1`. Добавление элементов в эту коллекцию осуществляется через переопределенный оператор побитового сдвига влево. В конструкторе класса, формирующего ответ, я добавляю элементы в эту коллекцию (`pages<<"/"<<"/index"<<"/files"<<"/files/add" << "/archiver" << "/archiver/remove" << "/archiver/create" << "/archiver/open";`), соответствующие нужным страницам. Далее, в методе отдающим код `html` конечному пользователю через прокладку в виде сокета и браузера, я реализую ветвление через оператор `case`. Именно возможность ветвления по строкам и игнорирование посторонних параметров побудило меня использовать эту коллекцию. Реализация этого ветвления (`switch(pages.indexOf(request)){ case 0:...`) предполагает получение индекса элемента из коллекции, что позволило решить проблему, убив двух зайцев. В ветвлении используются числа, которые эквивалентны порядковому номеру добавления элемента в коллекцию. А определение текущего элемента происходит через стандартный метод поиска совпадений `indexOf`. Благодаря этому я потратил минимум кода и последовал основному принципу объектно-ориентированного программирования, использованию существующего кода при написании последующего. В данном псевдокоде `request` – ссылка на запрашиваемую страницу. Если страница не существует, то будет выполнен код после ключевого слова `default`, который выведет отладочный код 404 (`default: response = "<h2>404</h2>"; break; }`). Стоит заметить, что в данном коде для генерации страницы ответа используется временная переменная `response`, а лишь затем происходит возвращение страницы. Это позволяет приравнять к этой переменной шаблон, а затем поменять в этой строке значения. Подобный

метод усложнит для хакера процесс замены шаблонов страниц, так как в процессе работы шаблоны константные, данные подставляются в копию шаблона в процессе генерации страницы-ответа.

В процессе работы программы необходимо получать со страниц информацию, введенную пользователем. Это необходимо для серфинга по каталогам, а именно для получения пути открываемого каталога. Реализовать это через ссылки было бы не правильно, так как это потребовало бы усложнения алгоритма поиска соответствия. Не стоит забывать, что сложность `QStringList` квадратичная, в первом цикле происходит проходка по элементам коллекции, а во втором проверка элементов коллекции на соответствие условию. Поэтому мной было решено написать функцию для последующей обработки строки. В библиотеке Qt не было готового решения, которое могло бы решить мою проблему. Продублирую для удобства ссылку с GET параметрами `http://127.0.0.1:1488/fm/show?folder=C:\Users`. Функция будет возвращать коллекцию вида “словарь” (`QMap<QString,QString> ret;`), где названию параметра будет соответствовать значение. Оба поля коллекции являются строками. И так, если в ссылке нет вопросительного знака, то функция вернет пустой словарь (`if(url.indexOf('?')== -1) return ret;`). Это означает, что в ссылке нет параметров. Далее, все содержимое ссылки до вопросительного знака будет урезано (`QString tmp = url.right(url.length()-url.indexOf('?')-1);`). Таким образом мы шаг за шагом подготавливаем строку к обработке стандартными средствами. После этого строка будет разделена на коллекцию подстрок, разделяемых символом `&` (`QStringList paramlist = tmp.split('&');`). Так в GET запросе разделяются параметры. А затем полученных список будет обработан в цикле, где левая часть подстроки отправится в адресную часть словаря, а правая – в содержимое (`for(int i=0;i<paramlist.count();i++){ QStringList paramarg = paramlist.at(i).split('='); ret.insert(paramarg.at(0),paramarg.at(1)); }`). После будет сделан отладочный вывод, а затем коллекция будет передана пользователю. Разбирать отладочный

вывод я не буду, так как это не имеет отношения к логике работы программы. После этого момента у нас готова системная часть программы, которая полноценно работает с латиницей.

Проблемы начинаются после использования русских букв. В процессе формирования GET запроса русские буквы заменяются на “Процентную кодировку”. Если перейти по ссылке снизу, откроется страница поиска google, на которой будут найдены сайты по слову “привет мир”
`http://google.com?q=%D0%BF%D1%80%D0%B8%D0%B2%D0%B5%D1%82+%D0%BC%D0%B8%D1%80`. Хотя, в GET параметр q была передана строка, состоящая из латинских букв, цифр и символов. Решить проблему удалось штатными средствами библиотеки Qt, добавив преобразование (`QString dir=QUrl::fromPercentEncoding(QByteArray::fromStdString(params.value("dir").toStdString())); dir.replace("+"," ");`). Так, конвертировав параметр мы получаем уже нормальное содержимое запроса, которое можно применить при работе с программой. Так-же в запросе знаки + заменяются на пробелы, так в “Процентной кодировке” указываются пробелы при передачи данных из веб форм. После этого можно уже перейти непосредственно к самой логической модели структуры страниц и параметров.

Всего программа реализует 7 страниц с различными параметрами, а именно /index – страница обертка для веб-движка Internet Explorer, созданная для предотвращения открытия страниц в новом окне, /files – список файлов, /files/add – добавление файлов в очередь к архивации, /archiver – список очереди к архивации, /archiver/remove – удаление страницы из очереди к архивации, /archiver/create – создание архива, /archiver/open – извлечение данных из архива. Страница со списком файлов(/files) выводит список файлов в директории. Она принимает параметр dir, который обозначает абсолютный путь к отображаемой папке. По умолчанию путь равен корневому каталогу файловой системы, в операционных системах серии Windows NT это диск C. В левой части страницы отображается список файлов в директории и подкаталоги. Любой подкаталог

можно открыть. При этом произойдет переход на новую страницу `/files`, а параметр `dir` будет задан как абсолютный путь к выбираемой директории. Файлы из списка можно добавить в очередь к архивации. При этом произойдет переход на страницу `/files/add` с параметром `file` равным абсолютному пути к файлу. На этой странице будет возможность снова открыть директорию с файлами, передав параметру `dir` путь к директории файла. Это позволит вернуться назад после добавления файла. После добавления файла на странице файлов при выводе содержимого каталога добавленные файлы будут помечены, их нельзя будет добавить в очередь снова. Файлы, которые являются результатом работы архиватора нельзя добавить в архив, их можно только извлечь. В правой части страницы находятся ссылки, которые открывают страницу списка архивации и запуска архивации. Далее я опишу страницу списка архивации.

Страница архивации (`/archiver`) содержит очередь архивации. Эта страница не получает параметров, в ней выводится исключительно список файлов. Зато в списке файлов можно нажать на ссылку для перехода на страницу `/archiver/remove`, что позволит удалить файл из списка архивации. Страница `/archiver/remove` получает параметр `file`, который равен абсолютному пути к файлу, который нужно удалить из списка архивации.

Страница `/archiver/create` открывается из страницы `/files`. Ссылка на нее находится в правой части таблицы и появляется, когда в списке архивации более одного файла. Страница получает параметр `path`, который должен содержать абсолютный путь к файлу, который должен создать архиватор из файлов в очереди архивации. После создания архива список очереди очищается. Файлы при этом не удаляются, просто создается архив в указанной директории. Архивация может быть удачной или неудачной. Архиватор предупредит, удалось ли создать архив.

Страница `/archiver/open` позволяет извлечь архив. Эта страница открывается при выборе архива, сделанного в данной программе, через список файлов

страницы /files. Данная страница получает параметр path. В текущем каталоге создается папка с псевдонимом копии программы, в которую идет извлечение. Если директория уже существует, извлечь архив не удастся. Это и есть отличительная особенность программы, извлечь два, созданных в одной программе, архива в одну папку нельзя, не смотря на название. Соответственно, легко будет понять, с какого компьютера файл. Эта принудительная стандартизация помешает студентам сдавать одни и те же файлы. Без этой системы создание нового архиватора было бы бессмысленно.

Теперь о архивации с точки зрения технической реализации. Архив представляет собой сжатый с использованием системы сжатия файлов zlib файл xml с набором тегов с атрибутами. Подобная организация не даст открыть этот файл сторонними средствами, формат архивации не похож ни на один существующий формат архивов на рынке. Файлы кодируются в base64 и записываются как атрибуты тегов <item>. Кроме этого, у тега <item> есть атрибут file и info. Если атрибут file задан, то item содержит файл и он извлекается из атрибута base64. Кроме этого, считывается имя файла из атрибута name. Если атрибут info задан, то item содержит системную информацию. Например, атрибут у тега info, который хранит псевдоним экземпляра программы именуется как appname. Так архиватор понимает, где был создан архив. Для разбора xml штатными средствами Qt все теги item помещаются в корневой тег root. Этот часть стандарта xml, игнорировать корневой тег нельзя. Пример содержимого файла xml без сжатия выглядит примерно так, <root><item file="true" base64="RGVmYXVsAgICAgICAgCcPu8u7i7iEK" name="AiOLog.txt"/><item info="true" appname="DEFAULTNAME"/></root>. В архиве обязательно содержится тег item с системной информацией. Количество файлов ограничено исключительно системными требованиями электронно-вычислительной машины. Имя файла при извлечение архива восстанавливаются не из base64, а из атрибута name. Это не даст студенту назвать файл нецензурным словом, названия файлов сохраняются. Кроме этого,

архиватор не фиксирует древо каталогов, из которых берутся файлы. Это упрощает работу преподавателя и создает стандарт структуры файла как отчета, я считаю, что так преподавателю будет удобнее проверять сданную работу.

- Архивы генерируются экземпляром класса `Archiver`. Этот класс реализует следующие методы: `GetList()` - позволяет получить копию списка путей к файлам для архивации, `IsFileInList()` - позволяет узнать, есть ли файл в списке, `AddFile()` - добавляет файл в список, `RemoveFile()` - позволяет удалить файл из списка, `CreateArchive()` - позволяет создать архив, `ExtractArchive()` - позволяет извлечь архив. Кроме этого, именно этот файл хранит коллекцию путей очереди архивации. Метод класса получает в параметры путь для создания архива и реализует запись кода xml, сжатие. Этот метод возвращает значение типа `bool`, обозначающее, удачно ли проведена архивация. Метод использует дружественную функцию `ReadBase64FromFile()`, которая получает прочитать содержимое файла как массив байтов и конвертировать его в `base64`.

Метод `ExtractArchive()` позволяет извлекать архивы. Именно он реализует логику извлечения архива, которую я описал выше. Он возвращает значение типа данных `bool`, обозначающее, удачно ли произошло извлечение. Метод использует для записи дружественную функцию `WriteBase64ToFile()`, которая реализует конвертацию из `base64` в бинарный формат и запись в файл. Функция возвращает `bool`, обозначающий, удалось ли совершить запись. Подобное решение тестировалось на текстовый файлов и не один символ не был поврежден. Решение оказалось эффективным, так как полностью решило поставленную задачу без нареканий. Не смотря на прирост в 30% при конвертации бинарного формата в текстовой, подобный изъян будет скомпенсирован сжатием через средства `zlib`, которое в некоторых случаях может быть более эффективным, чем простое сжатие файла без использования конвертации в `base64`.

`Base64` это специальный формат хранения файлов, который позволяет передавать файл как стоку через интернет или хранить небольшие файлы в

реляционной базе данных. В формате используются 64 символа из американской таблицы кодировки ASCII. Для того, чтобы преобразовать данные в base64, первый байт помещается в самые старшие восемь бит 24-битного буфера, следующий — в средние восемь и третий — в младшие значащие восемь бит. Если кодируется менее, чем три байта, то соответствующие биты буфера устанавливаются в ноль. Далее каждые шесть бит буфера, начиная с самых старших, используются как индексы строки «ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/» и её символы, на которые указывают индексы, помещаются в выходную строку. Если кодируются только один или два байта, в результате получаются только первые два или три символа строки, а выходная строка дополняется двумя или одним символами «=». Это предотвращает добавление дополнительных битов к восстановленным данным. Процесс повторяется над оставшимися входными данными. Кодирование Base64 может быть полезно, если в окружении HTTP используется информация, длину которой можно точно определить. Также многим приложениям необходимо кодировать двоичные данные для удобства включения в URL, скрытые поля форм, и здесь Base64 удобно не только для компактного представления, но и относительной нечитаемостью для попытки выяснения случайным человеком-наблюдателем природы данных.

Кодирование файлов в формат Base64 активно применяется в веб приложениях Google для интеграции ресурсов (а точнее, картинок) в каскадные таблицы стилей. Кроме этого, многие веб сайты преобразуют свои картинки в процессе вывода конечному пользователю, чтобы усложнить сохранение изображения (атрибут src у тега image не содержит ссылки на изображение, поэтому старые браузеры не могут сохранить это изображение). Это помогает защитить данные от копирования некоторой частью недоброжелательных пользователей, что в свою очередь формирует хорошую репутацию данного метода хранения файлов. С каждым днем это средство набирает популярность.

Кроме этого, программа фиксирует историю работы над файлами в СУБД. Поскольку подключение осуществляется через обертку в виде классов библиотеки Qt, можно динамично изменить конечное назначение данных путем выбора альтернативного драйвера подключения и строки подключения. Так-же, можно изменить таблицу назначения. Такой подход используется во многом современном ПО и позволяет расширить сферу работы программы в среде аппаратной поддержки. Кроме этого, можно подключить коммерческую базу данных (MySQL) на льготных условиях, так как сама база данных в комплекте поставки не устанавливается. Что в свою очередь позволяет использовать ПО под лицензией GPL. Данная лицензия позволяет использовать базу данных или любую другую программу, распространяемую под этой лицензией, свободно, но разработчик должен открыть покупателям исходный код. В большинстве случаев это сделать нельзя и тогда у владельца программы (в случае с MySQL на момент конца 2017 года это Oracle) покупается коммерческая лицензия. Так-же этот подход автоматически приводит возвращаемые значения базы данных в QString, что позволяет избежать проблем с кодировкой.

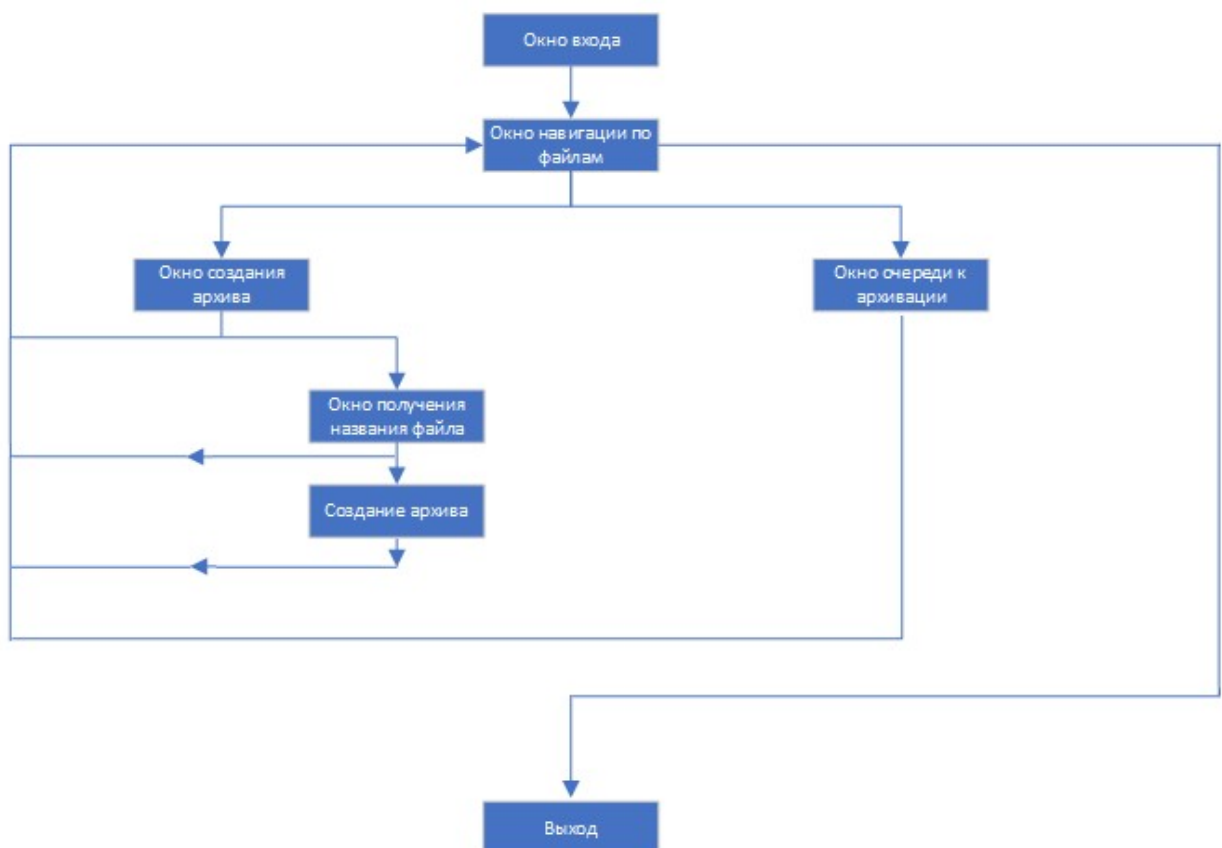
Проблема с кодировкой – одна из причин выбора подхода создания программы в виде веб сервера и логики. Стандартные методы ввода данных из консоли библиотеки STL считывают вводимый текст в кодировке ASCII. Но, к сожалению, в современных программах почти везде используется кодировка UTF-8. Например, в современных решениях Microsoft именно эта кодировка используется по умолчанию. Исправить проблему можно непосредственно вызовом метода SetContentCP() из Windows.h, но эта функция просто делает системный вызов и может не сработать из стороннего интерпретатора командной строки, подключенного через перенаправленный поток ввода-вывода. Другими словами, использование консоли в операционных системах серии Windows NT является не родным решением, в отличие от серии операционных систем Windows 9X, где оконный режим исключительно дополняет работу консоли. Кодировка строк по умолчанию в моей программе

устанавливается в функции `main()`, код команды выглядит так

```
QTextCodec::setCodecForLocale(QTextCodec::codecForName("UTF-8")).
```

Благодаря этому все константные строки шаблонов, ко которых я говорил в самом начале хранятся в кодировке UTF-8 и русские буквы на стороне клиента отображаются нормально. Так-же стоит заметить, что в заголовке `http` я указываю кодировку текста, `Content-Type: text/html; charset="utf-8"`. И это, именно `utf-8`. Эта кодировка позволяет эффективно передавать различные символы, в том числе и иероглифы, что позволяет использовать эту программу в китайских образовательных учреждениях. Так-же таблица символов этой кодировки содержит специальных символы-рисунки, которые можно использовать для оформления своей программы без загрузки картинок. Эти символы будут выглядеть нативно на любой платформе.

А ещё в функции `main()` я создаю экземпляр класса `QSettings` для считывания параметров из `ini` файла. Код выглядит так, `new QSettings(QDir::currentPath() + "/my_config_file.ini", QSettings::IniFormat);`. При этом в файл `my_config_file.ini` администратор пишет нужные настройки и именно там задается имя компьютера. Если файл не найден класс передаст в мой код вместо имени экземпляра программы заглушку `DEFAULTNAME` и программа все равно продолжит работать. Для разбора аргументов командной строки я использую класс `QCommandLineParser`. При запуске программы с аргументом `-i` программа автоматически создаст файл настроек и попытается создать нужные записи в таблице. Так-же в функции `main()` я создаю экземпляр класса `QSqlDatabase`, он подключается к базе данных. Далее создается экземпляр класса `ConnectionManager manager;`, который осуществляет мониторинг сокета. В самом низу функции `main()` запускается цикл событий, не дающий программе закрыться. Именно поэтому мониторинг сокета продолжается, не смотря на то, что в функции `main()` выполняемой стара строка с служебным словом `return`. Цикл событий в библиотеке Qt реализует класс `QApplication`, `QApplication a(argc, argv); ... return a.exec();` На этом теоретический разбор закончен.



Логическая модель программы.

2.4 ТЕСТИРОВАНИЕ ПРОГРАММЫ

Для тестирования программы кроме обычного режима архивации будут искусственно воссозданы нештатные режимы работы, не характерные для программы исходя из требований к функциональным характеристикам. Если вычислительная система и файлы соответствуют требованиям к надежности, то проблемы не должны возникнуть. Тем не менее, стоит понимать, что программа использует неуправляемый низкоуровневый код, поэтому стабильность её работы по определению уступает аналогам программ, использующих виртуальные машины. Сделано это было для удержания низкого ценового барьера для начала использования данного программного обеспечения: любой компьютер со средними характеристиками для современных образовательных учреждений на момент конца 2017 года сможет запустить программу. Однако проблема дает о себе знать, так как половина библиотеки Qt написана на языке Си, часть кода не выбрасывает исключений. Это не дает возможность получить любое исключение как экземпляр базового класса, что потребовало обработку известных на момент разработки нештатных ситуаций логическим ветвлением. На момент настоящего времени проблем возникать не должно, но при поддержке кода в будущем, возможно, потребуются вносить изменения в исходный код программы, добавляя операторы ветвления. Действительно, это основной минус применения языка системного программирования в прикладных целях, но множество других причин потребовало выбрать именно его. Более подробно об этом написано во введении в этом документе. Однако на подход к проведению тестирования это никак не влияет. Зато применение языка C++ позволило обеспечить достойный уровень производительности программы даже на электронной вычислительной системе, обладающей минимальными системными требованиями. Полученный результат можно охарактеризовать как положительный и удовлетворяющий условия поставленной задачи. Но, полностью сделать заключение о пригодности полученного программного обеспечения к употреблению массово можно только после тестирования.

№	Количество прогонок	Действие	Средний уровень сжатия	Комментарий	Результат
1	25	Попытка сжать 50 текстовых файлов, с русскими буквами и латиницей. После сжатия было произведено извлечение и проверка содержимого	40%	Использование текстовых файлов обеспечивает максимально возможную степень сжатия. В руководстве будет рекомендовано сохранять в архивы работы в текстовом формате CSV для таблиц и HTML для документов	Положительный
2	20	Попытка сжать 20 бинарных исполняемых файлов. После произведено извлечение и запуск.	-	Исполняемые файлы операционной системы Windows уже имеют степень сжатия и уменьшить их объем универсальным алгоритмом невозможно. Данные комплектуются в архив без сжатия	Положительный
3	10	Попытка сжать несколько архивов, созданных сторонним архиватором 7ZIP, в формате ZIP	-	Архивы, созданные в программе 7Zip уже имеют степень сжатия и поэтому добиться уменьшения объема информации универсальным алгоритмом невозможно. Данные комплектуются в архив без сжатия	Положительный
4	10	Попытка сжать несколько архивов, созданных сторонним архиватором 7ZIP, в формате 7Z	-	Архивы, созданные в программе 7Zip уже имеют степень сжатия и поэтому добиться уменьшения объема информации универсальным алгоритмом невозможно. Данные комплектуются в архив без сжатия	Положительный
5	10	Попытка сжать несколько архивов, созданных сторонним архиватором 7ZIP с максимально возможной степенью сжатия и словарем 900 кб, файла размером 1,44 мегабайта	-	Архивы, созданные в программе 7Zip уже имеют степень сжатия и поэтому добиться уменьшения объема информации универсальным алгоритмом невозможно. Данные комплектуются в архив без сжатия	Положительный
6	10	Попытка сжать несколько архивов, созданных сторонним архиватором 7ZIP, в формате Tar	-	Архивы, созданные в программе 7Zip уже имеют степень сжатия и поэтому добиться уменьшения объема информации универсальным алгоритмом невозможно. Данные комплектуются в архив без сжатия	Положительный
7	5	Попытка сжать несколько копий документа Microsoft Office Word 2003, содержащих черновики пояснительной записки к курсовому проекту	33,30%	Архиватор справился с поставленной задачей без потери данных	Положительный
8	1	Попытка сжать таблицу Excel из результатов предыдущих тестов	10%	Архиватор справился с поставленной задачей без потери данных	Положительный
9	1	Попытка создать архив из файлов, часть которых после добавления в очередь архивации была удалена	10%	Архив был создан. В процессе извлечения указанные файлы были записаны в архив как пустые. Остальная информация не повреждена	Положительный
10	1	Попытка извлечь архив в директорию с правами только на чтение	-	Архиватор оповестил пользователя о невозможности создания архива	Положительный
11	1	Попытка просмотреть список файлов в директории без прав на чтение	-	Архиватор отображает содержимое директории как пустое	Положительный
12	1	Попытка создать архив из файла образа диска установки реляционной базы данных MS SQL EXPRESS 64 разрядной версии	-	Архиватор был остановлен операционной системой, так как не было выделено нужное количество оперативной памяти	Отрицательный
Итого тестов: 95; Удачных: 94, Неудачных: 1					

Таблица 1. Результаты тестирования

Исходя из данных тестирования, представленных на таблице 1, можно вычислить коэффициент отказоустойчивости программы...

$$R(n) = 1 - \frac{n^-}{n} = 0,99$$

Надежность программы составляет 99%.

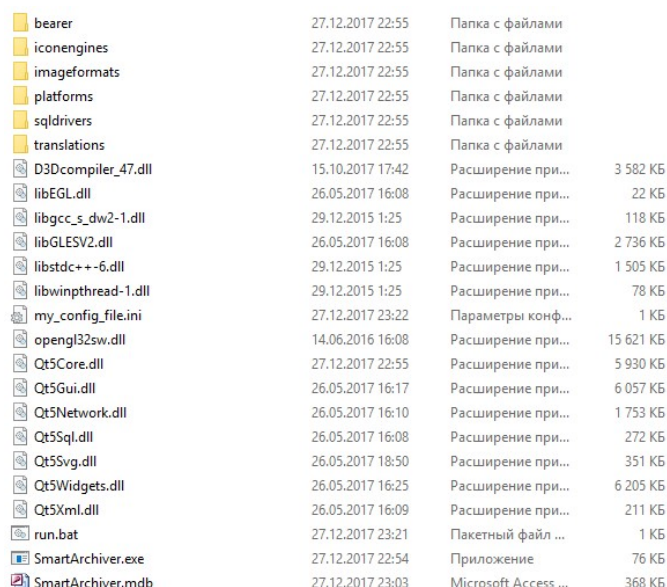
2.5 Анализ результатов тестирования

В результате тестирования было выяснено, что программа полностью удовлетворяет требованиям к программному продукту. Надежность программы составила 99%, что на 9% выше ожидаемых значений. При этом программа была экстренно остановлена в нештатных ситуациях операционной системой, а не внутренней ошибкой, когда ОС не могла выделить необходимые для продолжения работы ресурсы. Теоретически, изменив параметры операционной системы Windows, под которой было произведено тестирование, или, перекомпилировав программу под альтернативную операционную систему, поддерживаемую библиотекой Qt5, можно добиться промышленного качества программы, стремящегося к абсолютной работоспособности во всех условиях. Конечно, это требует детальной настройки каждой ЭВМ, что увеличит минимальных порог для начала использования данного программного обеспечения, поэтому в данное решение правки для достижения ещё большей отказоустойчивости вносить бессмысленно. Но возможность остается, что увеличивает множество задач, которые может решать данный архиватор. Так же в результате тестирования оказалось, что бинарный подход к разбору файла эффективнее текстового, если бы программа считывала файлы используя класс для текстового потока чтения, ей бы не удалось обработать бинарные файлы. Возможность укомплектовать исполняемые файлы в один, пусть даже больше в размере в разницу, равной сжатой системной информации о компьютере, где был создан архив, позволяет пересылать преподавателям не только документы, но и исполняемые файлы, собранные из исходных кодов программ. Это позволит использовать архиватор при отправке практических работ студентами преподавателям по электронной почте, студенты теряют анонимность при сдаче работ, а значит, вероятность преподавателя получить письмо с вредоносной программой падает. Невозможность сжимать исполняемые файлы вызвана тем, что в процессе компоновки объектных модулей современные компиляторы используют эффективные алгоритмы сжатия, нельзя сжать уже сжатый файл.

РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ

3.1. РУКОВОДСТВО ПРОГРАММИСТА

Программа предназначена для ведения работы с архивами в специфичном формате “.uffarch”. Программа должна запускаться от имени стороннего пользователя с специфичными правами, который должен иметь доступ на чтение и запись к файлу базы данных mdb и файлу конфигурации ini. Остальные пользователи должны иметь доступ к этим файлам в режиме “Только чтение”. Для использования этой программы необходим компьютер с операционной системой семейства Windows NT не ниже Windows 10, содержимое архива, в котором поставляется программа с настройками по умолчанию должно быть распаковано по пути C:\Projects\SmartArchiver



bearer	27.12.2017 22:55	Папка с файлами	
iconengines	27.12.2017 22:55	Папка с файлами	
imageformats	27.12.2017 22:55	Папка с файлами	
platforms	27.12.2017 22:55	Папка с файлами	
sqldrivers	27.12.2017 22:55	Папка с файлами	
translations	27.12.2017 22:55	Папка с файлами	
D3Dcompiler_47.dll	15.10.2017 17:42	Расширение при...	3 582 КБ
libEGL.dll	26.05.2017 16:08	Расширение при...	22 КБ
libgcc_s_dw2-1.dll	29.12.2015 1:25	Расширение при...	118 КБ
libGLESv2.dll	26.05.2017 16:08	Расширение при...	2 736 КБ
libstdc++-6.dll	29.12.2015 1:25	Расширение при...	1 505 КБ
libwinpthread-1.dll	29.12.2015 1:25	Расширение при...	78 КБ
my_config_file.ini	27.12.2017 23:22	Параметры конф...	1 КБ
opengl32sw.dll	14.06.2016 16:08	Расширение при...	15 621 КБ
Qt5Core.dll	27.12.2017 22:55	Расширение при...	5 930 КБ
Qt5Gui.dll	26.05.2017 16:17	Расширение при...	6 057 КБ
Qt5Network.dll	26.05.2017 16:10	Расширение при...	1 753 КБ
Qt5Sql.dll	26.05.2017 16:08	Расширение при...	272 КБ
Qt5Svg.dll	26.05.2017 18:50	Расширение при...	351 КБ
Qt5Widgets.dll	26.05.2017 16:25	Расширение при...	6 205 КБ
Qt5Xml.dll	26.05.2017 16:09	Расширение при...	211 КБ
run.bat	27.12.2017 23:21	Пакетный файл ...	1 КБ
SmartArchiver.exe	27.12.2017 22:54	Приложение	76 КБ
SmartArchiver.mdb	27.12.2017 23:03	Microsoft Access ...	368 КБ

Рисунок 1. Древо каталогов программы.

Для запуска программы необходимы динамически подключаемые модули библиотеки Qt. В стандартном комплекте поставки они включены. Если один из модулей будет утерян, запуск программы будет невозможен и её придется переустанавливать.

Кроме этого, на компьютере должен быть установлен Microsoft Access или другая реляционная база данных, способная использовать драйвер ODBC. Если планируется использовать альтернативную базу данных, а не Access, то файл

SmartArchiver.mdb можно удалить, а после отредактировать файл my_config_file.ini и указать новую строку подключения. Примерное содержимое файла конфигурации программы находится снизу.

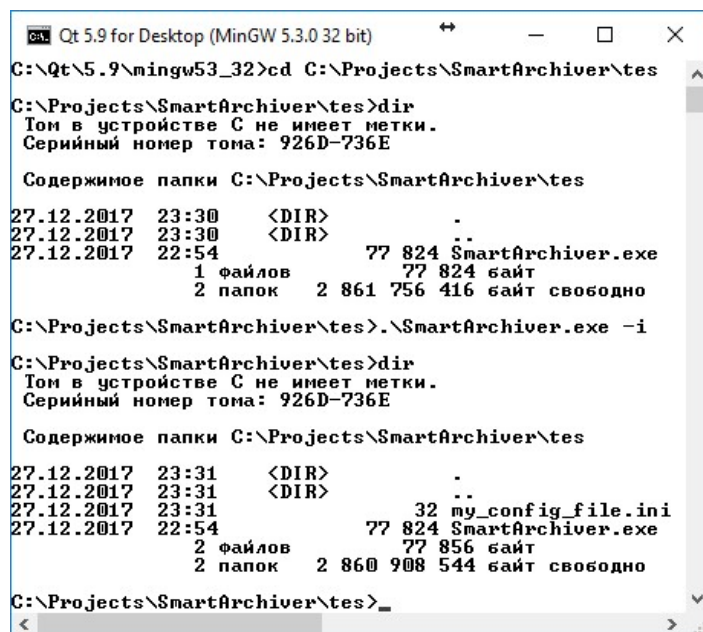
[General]

AppName=DEFAULTNAME

DRIVER=QODBC

ConnectionString=Новая строка подключения

Так же можно динамически сгенерировать файл настроек архиватора, запустив его с параметром `-i`



```
Qt 5.9 for Desktop (MinGW 5.3.0 32 bit)
C:\Qt\5.9\mingw53_32>cd C:\Projects\SmartArchiver\tes
C:\Projects\SmartArchiver\tes>dir
Том в устройстве C не имеет метки.
Серийный номер тома: 926D-736E

Содержимое папки C:\Projects\SmartArchiver\tes
27.12.2017 23:30 <DIR> .
27.12.2017 23:30 <DIR> ..
27.12.2017 22:54 77 824 SmartArchiver.exe
                1 файлов 77 824 байт
                2 папок 2 861 756 416 байт свободно

C:\Projects\SmartArchiver\tes>.\SmartArchiver.exe -i
C:\Projects\SmartArchiver\tes>dir
Том в устройстве C не имеет метки.
Серийный номер тома: 926D-736E

Содержимое папки C:\Projects\SmartArchiver\tes
27.12.2017 23:31 <DIR> .
27.12.2017 23:31 <DIR> ..
27.12.2017 23:31 32 my_config_file.ini
27.12.2017 22:54 77 824 SmartArchiver.exe
                2 файлов 77 856 байт
                2 папок 2 860 908 544 байт свободно

C:\Projects\SmartArchiver\tes>
```

Рисунок 2. Динамическая генерация файла конфигурации

Если строка подключения указана неверно, то вы получите уведомление о невозможности подключения к базе данных. Его рисунок представлен ниже

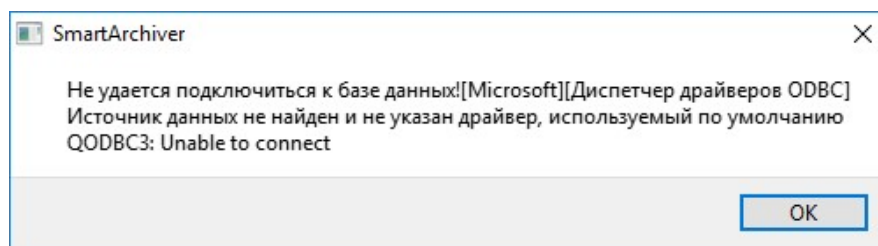


Рисунок 3. Неверно указана строка подключения.

В один момент времени может быть запущена только одна копия архиватора. Для предотвращения двухкратного запуска используйте для старта специальный файл Run.bat, содержащий сценарий консольных команд Windows для правильного запуска. Если в процессе запуска появляется подобное диалоговое окно, как на рисунке снизу

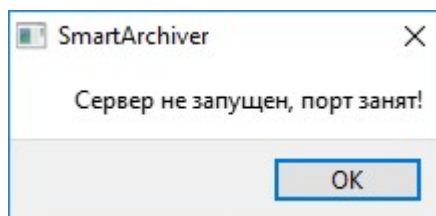


Рисунок 4. Порт занят.

То это значит, что программа не может получить доступ к порту 1488 для отрисовки пользовательского интерфейса в интерпритатор веб приложений Microsoft MSHTA. Порт 1488 нужно освободить.

Выходной файл программы представляет собой сжатый, актуальным только для этого программного обеспечения, архив, содержащий помимо файлов информацию о копии программы. Из файла конфигурации можно задать псевдоним для каждой копии программы. Благодаря этому появляется возможность вести аудит. Аудит представляет собой реляционную базу данных с доступом к данным через язык запросов SQL, предоставляющую доступ к двум таблицам

Действия	
Поле	Тип
Код	Счетчик
Название	nvarchar(255)

История	
Поле	Счетчик
Имя файла	nvarchar(255)
Имя архива	nvarchar(255)
Имя пользователя	nvarchar(255)
Действие	Числовой

Таблица 2. Структура базы данных.

При этом содержимое таблицы действия статично. Запись с индексом 1 должна обозначать создание архива, с индексом 2 – удаление. Только так, создав ту же структуру таблиц, вы сможете подключить архиватор к альтернативной базе данных. Примерное содержание заполненной таблицы История смотрите ниже.

Код	Имя файла	Имя архива	Имя пользс	Действие	Щелкните для добавления
1	/tmp/Фа.txt	/tmp/undefin	DEFAULTNAME	Создание	
2	/tmp/DEFAULT	/tmp/undefin	DEFAULTNAME	Извлечение	
*	(№)				

Рисунок 5. Примерное содержание таблицы.

К подобной таблице можно применить несколько запросов. В базе данных Access представлены следующие запросы:

1. Получить список пользователей, создававших архивы

```
SELECT DISTINCT История.[Имя пользователя], История.[Действие]
FROM История
WHERE История.[Действие]=1;
```

2. Получить список пользователей, извлекавших архивы

```
SELECT DISTINCT История.[Имя пользователя], История.[Действие]
FROM История
WHERE История.[Действие]=2;
```

3. Узнать, кто работал с архивом

```
SELECT История.[Имя архива], История.[Имя пользователя]
FROM История
WHERE (((История.[Имя архива])=[Введите путь к архиву]));
```

Используя подобные SQL запросы можно вести аналитику и проверять, какие архивы были созданы студентами. Поэтому выходной информацией данного ПО является не только сами архивы, но и создаваемая в процессе использования база данных. Это и есть отличительная особенность данного программного средства. Именно эта функция позволяет выбрать архив как основной инструмент при обработке передаваемой информации между студентами и преподавателями. Фиксирование всех действий над архивами позволяет обеспечить индивидуальность работ, обезопасить сеть компьютеров от вредоносного программного обеспечения, а так же упростить работу администратора при выявлении виновных программных поломок электронной вычислительной системы.

3.2 Руководство пользователя

Эта программа создана для сжатия студентами файлов практических работ для последующей отправки преподавателям. Основной особенностью программы является полная отчетность перед администрацией о сжимаемых и разархивируемых файлах для обеспечения индивидуальности каждой работы. Архиватор использует свой формат сжатия файлов, поэтому избежать добавления записи в историю архивации невозможно.

Подразумевается, что системный администратор настроит все копии программы таким образом, чтобы они отправляли историю архивации на удаленный сервер. Но по умолчанию архиватор помещает историю в локальную файловую базу данных Microsoft Access. На одном компьютере может быть установлено несколько копий данного архиватора, по одной копии на каждого пользователя. Но в один момент может быть открыта только одна. В документации для программиста описан процесс запуска архиватора, но выбор между запуском после клика по ярлыку и автозагрузкой остается на стороне системного администратора.

После запуска программы открывается окно входа. Тут пользователь может проверить, от чьего имени идет работа с архивами. Сделано это на случай, если другой пользователь забыл выйти из системы и была открыта его копия архиватора.

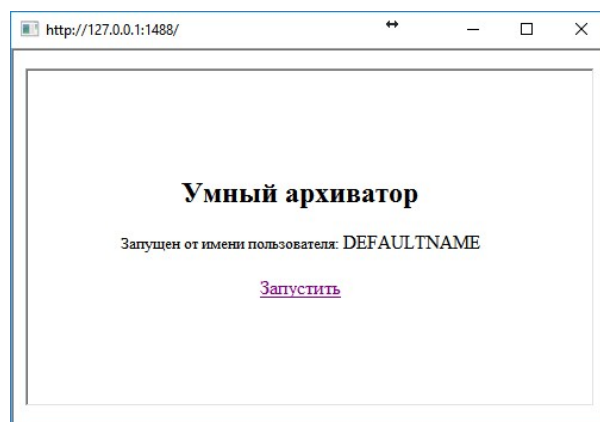


Рисунок 6. Окно входа

После нажатия на кнопку запустить пользователь попадает в меню выбора файлов. Оно состоит из части управления(справа) и части обзора файловой системы(слева), отделенных вертикальной чертой. Изначально обзор файлов начинается с корневого каталога диска С.

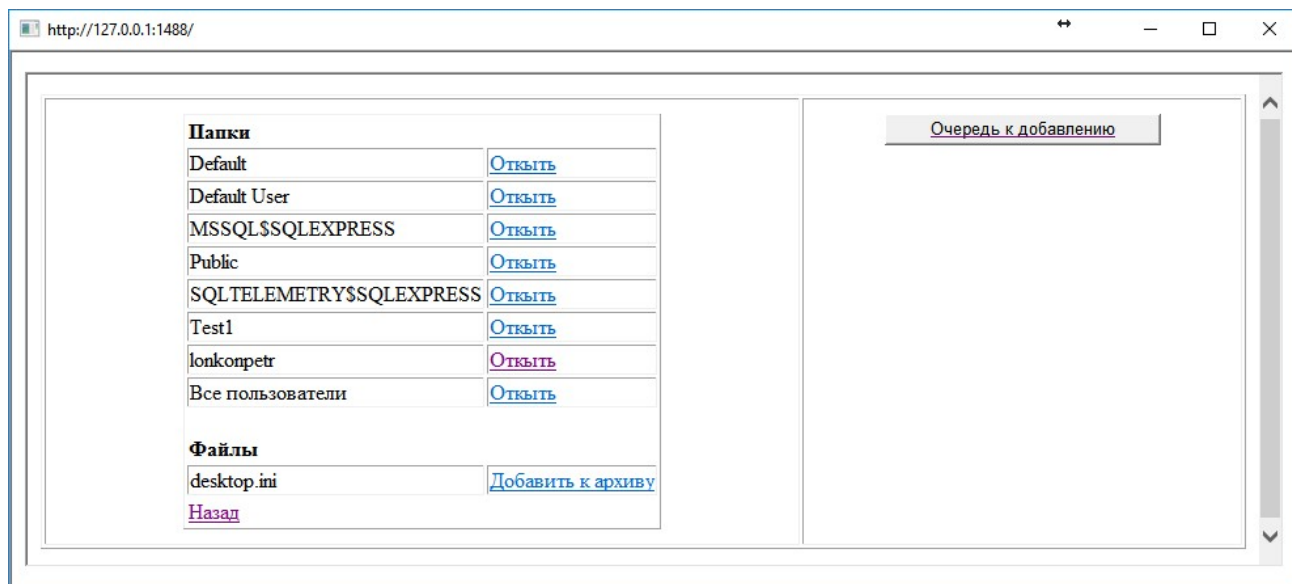


Рисунок 6. Выбрать файлы.

Этот архиватор не сохраняет структуру директорий в процессе архивации. В архиве все файлы содержатся на одном уровне. Поэтому в поле обзора файловой системы папки можно только открыть и они показываются сверху. Снизу идут файлы, лежащие в текущей директории.

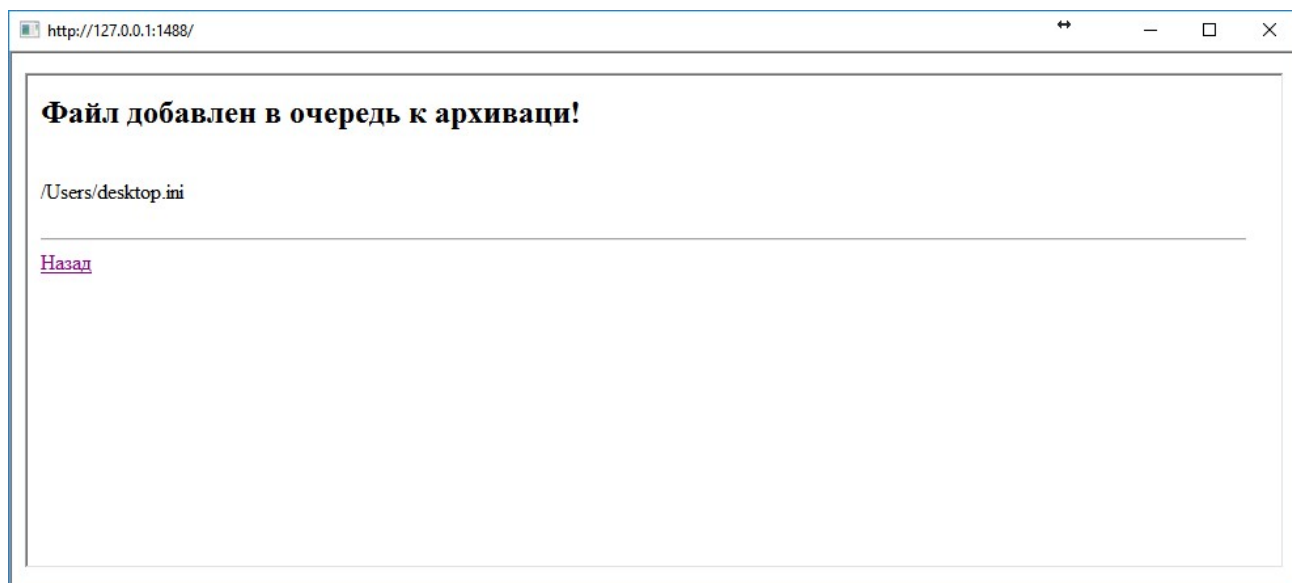


Рисунок 6. После добавления файла.

После нажатия на интерактивную кнопку с надписью “Добавить к архиву”, используя манипулятор типа “Мышь” путь к файлу записывается в очередь к архивации. На этом этапе стоит обеспечить полный доступ на чтение и запись к файлу, закрыв его во всех открытых программах. Не стоит удалять файл.

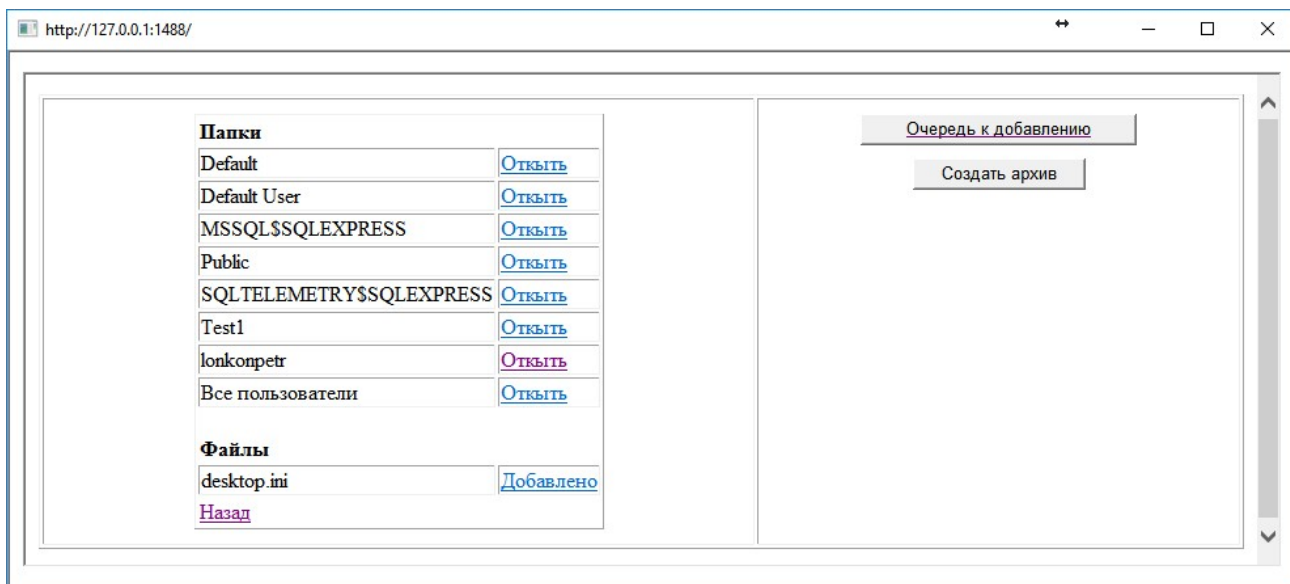


Рисунок 7. После нажатия на кнопку “Назад”.

После нажатия на кнопку с текстом “Назад” в меню справа появляется кнопка создать архив. Она появляется как только в очередь к архивации добавляется хотя бы один файл. По аналогии можно выбрать несколько файлов. Обратите внимание, что повторно выбрать файлы не получится...

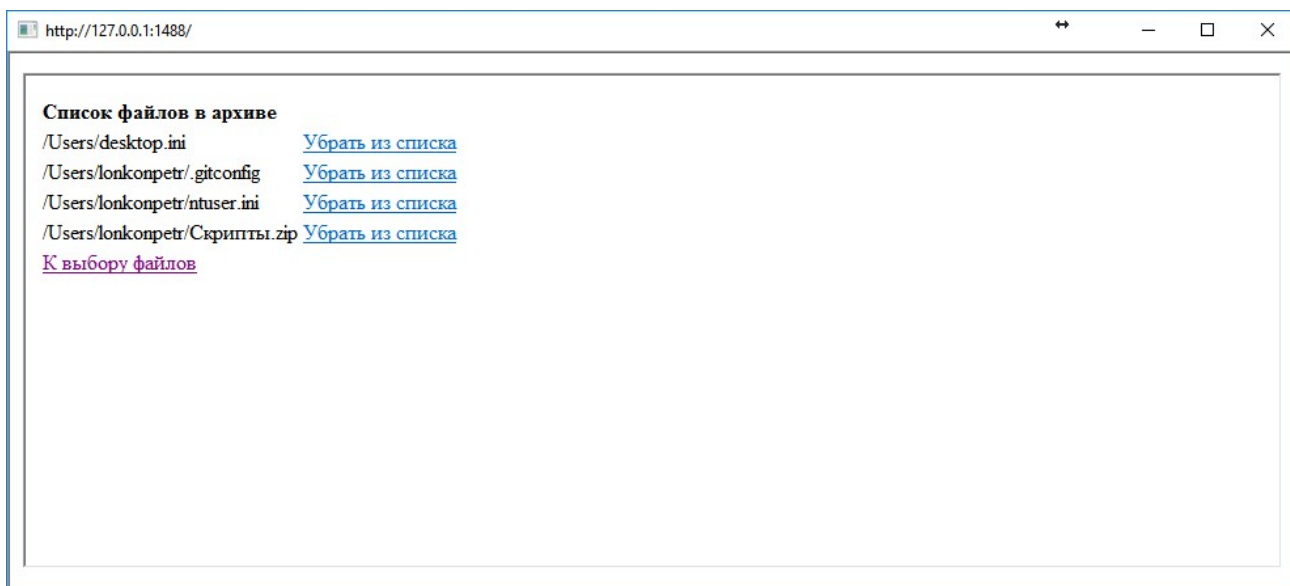


Рисунок 8. Очередь к архивации.

Если один из файлов был выбран ошибочно, его можно убрать из очереди. Для этого перейдите в очередь к архивации, используя кнопку с текстом “Очередь к добавлению”, Появится список файлов, которые находятся в очереди.

Используя кнопку “Удалить из списка” вы можете удалить файл из очереди.

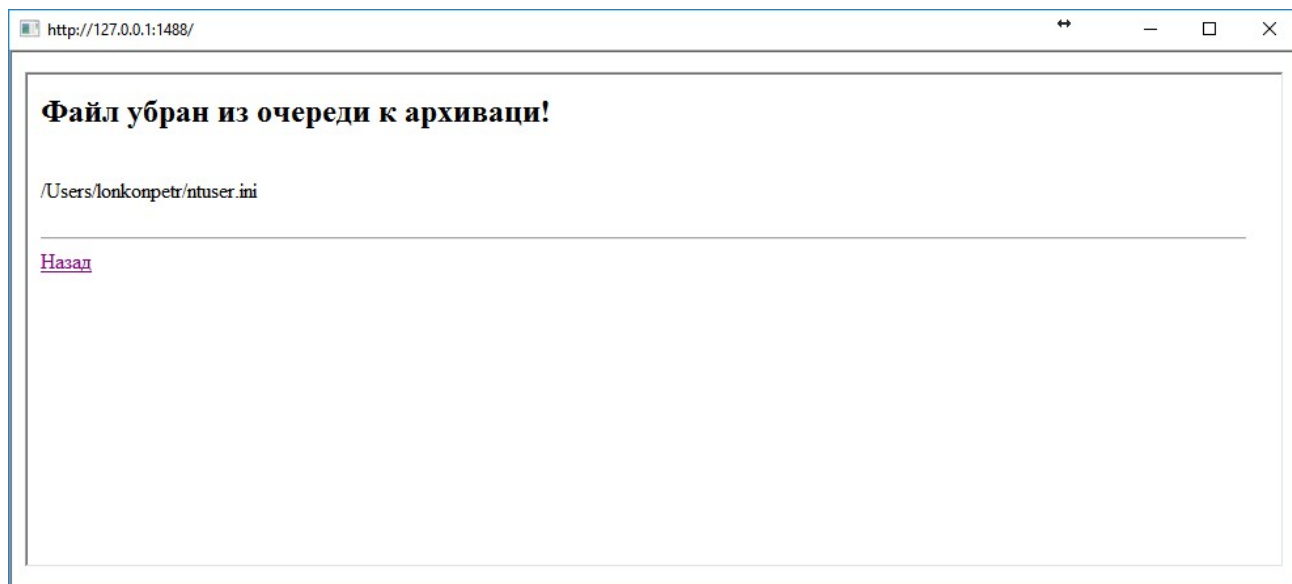


Рисунок 9. Убрать из очереди

Далее, используя кнопку с текстом “Назад”, а затем кнопку с текстом “К выбору файлов” можно вернуться к обзору файловой системы. На странице обзора файловой системы нажмите в каталоге, в котором хотите создать архив, кнопку с текстом “Создать архив”

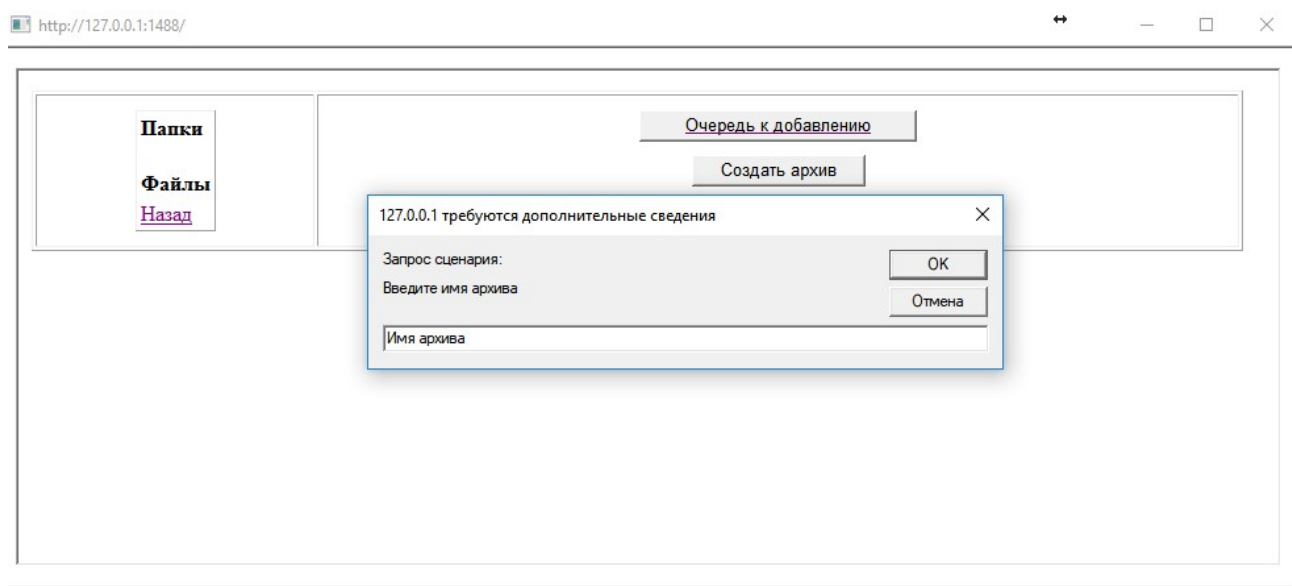


Рисунок 10. Создание архива

Вам будет предложено ввести имя архива. После ввода архива в выбранной директории будет создан новый файл с расширением “uffarch”. Это и есть созданный архив. Обратите внимание, что архиватор не предлагает добавить этот архив в очередь к сжатию, не смотря на то, что это тоже файл. Архиватор может только его разархивировать.

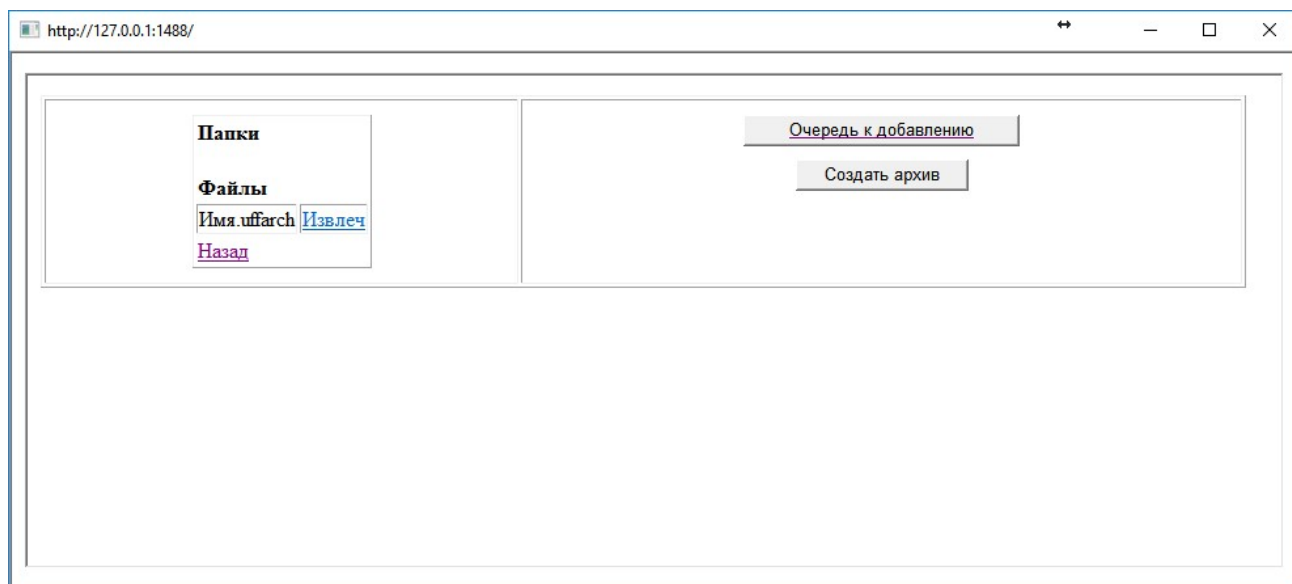


Рисунок 11. Созданный архив

После нажатия на кнопку с текстом “Извлечь” создается папка, соответствующая имени пользователя. Одновременно в одну директорию может быть извлечен только один архив, созданных одним пользователем. В сочетании с системой ведения истории это и позволит преподавателям определить индивидуальность отправленной ему студентом работы.



Рисунок 11. После извлечения архива

Все файлы, добавленные в архив, сохраняются в созданной директории. Так преподавателю сразу будет видно количество файлов и их размер



```
Командная строка
c:\tmp>tree /f
Структура папок
Серийный номер тома: 000000C6 926D:736E
C:\tmp
├── Имя.uffarch
└── DEFAULTNAME
    ├── .gitconfig
    ├── desktop.ini
    └── Скрипты.zip

c:\tmp>
```

Рисунок 12. Полученная директория

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы были достигнуты все поставленные цели и задачи. Была создана программа, написанная на языке программирования C++, которая может помочь преподавателям избавиться от списывания на практических работах, выполняемых на компьютере.

Была создана система учета архивации и извлечения файлов, которая поможет фиксировать неправомерный доступ к информации при использовании данного архиватора.

Исходный код данной программы прост для понимания и позволяет с легкостью создать на его основе новые кроссплатформенные приложения с графическим пользовательским интерфейсом.

При разработке данной программы мной было изучено много технологий, связанных как с прикладным программированием, так и с системным. Полученный мной опыт я буду использовать в своей дальнейшей учебе и научной деятельности. Мной был изучен протокол HTTP, я научился с ним работать. Кроме этого, полученный графический фреймворк можно достать из программы и использовать повторно, так как в процессе разработки я применил возможности ООП и организовал код так, чтобы часть, отвечающая за отрисовку графики была в отдельном модуле.

Кроме этого, программа работает с реляционными базами данных. На примере программы можно увидеть, как в современных языках программирования организована работа с большими объемами данных.

С уверенностью могу сказать, что основное достоинство программы – её огромная сфера применения. Действительно, отсутствие возможности удостовериться, кто автор архива, является основным недостатком архиваторов, существующих на рынке

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Стандарты

1. ГОСТ 7.32. – 2001. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21с.
2. ГОСТ 7.82 -2001. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления – М.: ИПК Издательство стандартов, 2001. – 21 с.
3. Единая система программной документации. – М.: СТАНДАРТИНФОРМ, 2005.

Монографии, учебники, учебные пособия

4. Макс Шлее Qt 5.3. Профессиональное программирование на C++. 2015. –928 с.

ПРИЛОЖЕНИЕ А

Исходный код программы

Файл main.cpp

```
#include <QApplication>
```

```
#include <QtNetwork/QTcpServer>
```

```
#include <QtNetwork/QTcpSocket>
```

```
#include <QObject>
```

```
#include <QDebug>
```

```
#include <QMessageBox>
```

```
#include <QUrlQuery>
```

```
#include <QUrl>
```

```
#include <QMap>
```

```
#include <QFile>
```

```
#include <QTextCodec>
```

```
//Инициализация QSql
```

```
#include <QSqlDatabase>
```

```
#include <QSqlQuery>
```

```
#include <QSqlError>
```

```
//Работа с файлами
```

```
#include <QDirIterator>
```

```
#include <QDir>
```

```
#include <windows.h>
```

```
#include <QProcess>
```

```
#include <QFileDialog>
```

```
#include <QDomDocument>
```

```
#include <QSettings>
```

```
#include <QCommandLineParser>
```

```
//Общедоступный объект, осуществляющий подключение к базе данных
```

```
QSqlDatabase db;
```

```

//Объект настроек проекта, получающий данные из ini файлы
QSettings* settings;
//Считывает файл в base64
QString ReadBase64FromFile(QString path,QString archpath)
{
    QFile f(path);
    f.open(QIODevice::ReadOnly);
    QString ret = f.readAll().toHex();
    QSqlQuery query;
    query.exec("INSERT INTO История([Имя файла], [Имя архива], [Имя
пользователя], [Действие]) VALUES ('"+path+"','"+archpath+"','"+settings-
>value("AppName", "DEFAULTNAME").toString()+",1)");
    f.close();
    return ret;
}
bool WriteBase64ToFile(QString base64, QString path, QString archpath)
{
    QFile f(path);
    f.open(QIODevice::WriteOnly);
    QByteArray extract;
    extract.append(base64);
    f.write(QByteArray::fromHex(extract));
    QSqlQuery query;
    query.exec("INSERT INTO История([Имя файла], [Имя архива], [Имя
пользователя], [Действие]) VALUES ('"+path+"','"+archpath+"','"+settings-
>value("AppName", "DEFAULTNAME").toString()+",2)");
    f.close();
    return true;
}

```



```

//Функция для разбора GET параметров у ссылки
QMap<QString,QString> ParseUrlParameters(QString &url)
{
    //Заранее создаем коллекцию для возвращаемого значения
    QMap<QString,QString> ret;
    //Параметров нет. Возвращаем пустую коллекцию
    if(url.indexOf("?")==-1)
    {
        return ret;
    }
    //Обрезаем из url все до вопроса
    QString tmp = url.right(url.length()-url.indexOf("?")-1);
    //Разбиваем в коллекцию параметров через разделитель &
    QStringList paramlist = tmp.split('&');
    //Ввожу параметры в коллекцию
    for(int i=0;i<paramlist.count();i++)
    {
        QStringList paramarg = paramlist.at(i).split('=');
        ret.insert(paramarg.at(0),paramarg.at(1));
    }
    //Вывожу в консоль полученные параметры
    QMapIterator<QString, QString> i(ret);
    while (i.hasNext()) {
        i.next();
        qDebug() << i.key() << ":" << i.value() << endl;
    }
    return ret;
}

```

//Этот класс содержит список файлов и архивирует файлы. Делает вывод

```
class Archiver{
private:
    QList<QString> filelist;
public:
    Archiver()
    {
    }
    QList<QString> GetList()
    {
    return filelist;
    }
    bool IsFileInList(QString file)
    {
    for(int i=0;i<filelist.count();i++){
    if(filelist.at(i)==file) return true;
    }
    return false;
    }
    void AddFile(QString file)
    {
    filelist.append(file);
    }
    void RemoveFile(QString file)
    {
    for(int i=0;i<filelist.count();i++)
    {
    if(filelist.at(i)==file){
    filelist.removeAt(i);
```

```

return;
}
}
qDebug() << "Файла не существует";
}
bool CreateArchive(QString filename){
QDomDocument xml;
QDomElement root=xml.createElement("root");
xml.appendChild(root);
//добавление файла в архив
for(int i =0; i<filelist.count();i++)
{
QDomElement node=xml.createElement("item");
node.setAttribute("file","true");
node.setAttribute("name",QFileInfo(filelist[i]).fileName());
node.setAttribute("base64",ReadBase64FromFile(filelist[i],filename));
root.appendChild(node);
}
//Добавление имени пользователя
QDomElement name=xml.createElement("item");
name.setAttribute("info","true");
name.setAttribute("appname",settings-
>value("AppName","DEFAULTNAME").toString());
root.appendChild(name);
QFile file1(filename);
if(!file1.open(QIODevice::WriteOnly))
{
return false;
}
}

```

```

else
{
file1.write(qCompress(xml.toByteArray()));
file1.close();
}
return true;
}
bool ExtractArchive(QString path)
{
QFile f(path);
f.open(QIODevice::ReadOnly);
if(!f.isOpen()) return false;
QDomDocument doc;
if (!doc.setContent(qUncompress(f.readAll())) {
f.close();
return false;
}
f.close();
//Получить имя пользователя, создавшего архив
QString name;
QDomNode root = doc.firstChild();
for(int i=0;i<root.childNodes().count();i++)
{
if(root.childNodes().at(i).attributes().contains("info"))
{
name=root.childNodes().at(i).toElement().attribute("appname","-1");
}
}
if(QDir(QFileInfo(path).dir().path()+"/"+name).exists()) return false;

```

```

QFileInfo(path).dir().mkdir(name);
for(int i=0;i<root.childNodes().count();i++)
{
if(root.childNodes().at(i).attributes().contains("file"))
{
WriteBase64ToFile(root.childNodes().at(i).toElement().attribute("base64","1"),QFileI
nfo(path).dir().path()+"/"+name+"/"+root.childNodes().at(i).toElement().attribute("na
me","-1"),path);
}
}
return true;
}
}archiver;

//Этот класс собирает код html для ответа клиенту по текущей ссылке
class HtmlResponse{
private:
//Перечисление страниц. Страница должна начинаться с /. Допустимо слитное
написание для подстраниц
QStringList pages;
//HTML код главной страницы
const QString mainpage="<table width='100%' height='100%'><tr><td
align='center'>"
"<h2>Умный архиватор</h2>"
"<span><small>Запущен от имени пользователя:</small>"
"{USERNAME}</span><br><br>"
"<a href='/files?dir=' target='_self'>Запустить</a>"
"</td></tr></table>";
const QString infopage="<h2>{TITLE}</h2><br><p>{INFO}</p><hr><a
href='{BACKURL}'>Назад</a>";

```

```

public:
    HtmlResponse()
    {
        //Запись страниц
        pages<<"/"<<"/index"<<"/files"<<"/files/add" << "/archiver" << "/archiver/remove"
        << "/archiver/create" << "/archiver/open";
    }
    QString GetHtml(QString request)
    {
        QMap<QString,QString> params=ParseUrlParameters(request);
        //Параметры были. Для упрощенной обработки их стоит удалить
        if(params.count()!=0)
        {
            request.remove(request.indexOf('?'),request.length()-request.indexOf('?'));
        }
        qDebug() << "Запрос на страницу " <<request;
        QString response;
        switch(pages.indexOf(request))
        {
            //Главная страница. Выводит фрейм
            case 0:
            {
                response = "<html><head><HTA:APPLICATION APPLICATIONNAME='oHTA'
                scroll='no' ID='oHTA' VERSION='1.0'/></head><body>"
                "<iframe src=\"http://127.0.0.1:1488/index\" width=\"100%\"
                height=\"100%\"application=\"yes\"> "
                "</body></html>";
            }
        }
        break;
    }

```

```

//Фрейм меню выбора страницы
case 1:
{
response = mainpage;
response.replace("{USERNAME}",settings-
>value("AppName","DEFAULTNAME").toString());
}
break;
//Диспетчер файлов
case 2:
{
QString
dir=QUrl::fromPercentEncoding(QByteArray::fromStdString(params.value("dir").toS
tdString()));
dir.replace("+"," ");
response+="




```

```

href='/files?dir="+dir+"/"+folders.at(i)+">Открыть</a></td></tr>";
}
response+="<tr><td style='border: 0px solid transparent;'><br><p style='font-weight:
bold;'>Файлы</p></td></tr>";
QStringList files = directory.entryList(QDir::NoDotAndDotDot | QDir::System
QDir::Hidden | QDir::Files, QDir::DirsFirst);
for(int i=0;i<files.count();i++) {
//Этот файл- наш архив. Подобные не архивируем, а предлагаем извлечь
if(QFileInfo(files[i]).suffix()=="uffarch")
{
response+="<tr><td>"+files.at(i)+"</td><td><a
href='/archiver/open?path="+dir+"/"+files.at(i)+">Извлечь</a></td></tr>";
}
else
{
QString addurl = (archiver.IsFileInList(dir+"/"+files.at(i)))?"<a
href='#>Добавлено</a>':"<a href='/files/add?file="+dir+"/"+files.at(i)+">Добавить
к архиву</a>";
response+="<tr><td>"+files.at(i)+"</td><td>"+addurl+"</td></tr>";
}
}
QDir updirectory(directory.path());
updirectory.cdUp();
response+="<tr><td style='border: 0px solid transparent;'><a
href='/files?dir="+updirectory.path()+">Назад</a></td></tr>";
response+="</table>";
response+="</td><td style='vertical-align: top;' align='center'><a
href='/archiver'><button onclick='window.location.href=\"/archiver\";' style=' margin-
top: 10px;'>Очередь к добавлению</button></a></br>";

```



```

if(archiver.GetList().count()>0)response+="<button
onclick='window.location.href=\"/archiver/create?path="+directory.path()+"^\""+w
indow.prompt(\"Введите имя архива\")+\".uffarch\";' style=' margin-top:
10px;'>Создать архив</button>";
response+="</td></tr></table>";
}
break;
//Добавление файла в архив
case 3:
{
//Преобразование из "процентного вида <form>" в человеческий вид
QString
file=QUrl::fromPercentEncoding(QByteArray::fromStdString(params.value("file").to
StdString()));
file.replace("+"," ");
qDebug() << "Добавление файла "+file;
archiver.AddFile(file);
QFileInfo fileinfo(file);
response = infopage;
response = response.replace("{TITLE}","Файл добавлен в очередь к
архивации!").replace("{INFO}",file).replace("{BACKURL}","/files?dir="+fileinfo.di
r().path());
}
break;
//Список очереди к архивации
case 4:
{
QList<QString> filelist= archiver.GetList();
response+="<table>";

```

```

response+="<tr><td><p style='font-weight: bold;'>Список файлов в
архиве</p></td></tr>";
for(int i=0;i<filelist.count();i++)
{
response+="<tr><td>"+filelist.at(i)+"</td><td><a
href='/archiver/remove?file="+filelist.at(i)+">Убрать из списка</a></tr>";
}
response+="<tr><td><a href='/files?dir=/'>К выбору файлов</a></td></tr>";
response+="</table>";
}
break;
//Убрать файл из очереди
case 5:
{
//Преобразование из "процентного вида <form>" в человеческий вид
QString
file=QUrl::fromPercentEncoding(QByteArray::fromStdString(params.value("file").to
StdString()));
file.replace("+"," ");
qDebug() << "Удаление файла "+file;
archiver.RemoveFile(file);
QFileInfo fileinfo(file);
response = infopage;
response = response.replace("{TITLE}","Файл убран из очереди к
архивации!").replace("{INFO}",file).replace("{BACKURL}","/archiver");
}
break;
//Создать архив
case 6:

```

```

{
QString
path=QUrl::fromPercentEncoding(QByteArray::fromStdString(params.value("path").
toStdString()));
path.replace("+"," ");
response=infopage;
if(archiver.CreateArchive(path))response = response.replace("{TITLE}","Файл
создан!").replace("{INFO}","Файл успешно
создан").replace("{BACKURL}","/files?dir="+QFileInfo(path).dir().path());
else response = response.replace("{TITLE}","Файл не
создан!").replace("{INFO}","Не удастся создать
архив...").replace("{BACKURL}","/files?dir=/");
}
break;
//Извлеч архив
case 7:
{
QString
path=QUrl::fromPercentEncoding(QByteArray::fromStdString(params.value("path").
toStdString()));
path.replace("+"," ");
response=infopage;
if(archiver.ExtractArchive(path))response = response.replace("{TITLE}","Архив
извлечен!").replace("{INFO}","Файлы успешно
созданы").replace("{BACKURL}","/files?dir="+QFileInfo(path).dir().path());
else response = response.replace("{TITLE}","Не удастся извлеч
архив!").replace("{INFO}","Не удастся извлечь данные из
архива...").replace("{BACKURL}","/files?dir=/");
}
}

```

```

break;
default:
response = "<h2>404</h2>";
break;
}
return response;
}
}ResponseManager;
//Этот класс осуществляет низкоуровневую обработку запросов к серверу
class ConnectionManager
{
private:
QTcpServer *tcpServer;
public:
ConnectionManager()
{
tcpServer = new QTcpServer();
//Ожидание запроса на подключение
QObject::connect(tcpServer,QTcpServer::newConnection,[this]()
{
qDebug()<<"Новое подключение!";
//Выделение сокета и передача управления обработчику
QTcpSocket* tempClientSocket=tcpServer->nextPendingConnection();
QObject::connect(tempClientSocket,QTcpSocket::readyRead,[tempClientSocket,this]
){
//Считывание запроса
QString request;
if(tempClientSocket->bytesAvailable())
{

```

```

qDebug() << "Чтение сокета";
QString tmp= tempClientSocket->readAll();
request=tmp.split(" ").at(1);
qDebug() << "Считано: "<<request;
}
else
{
qDebug() << "Сокет отброшен: данные не получены!";
tempClientSocket->close();
} //Вывод ответа клиенту
QTextStream os(tempClientSocket);
os.setAutoDetectUnicode(true);
os << "HTTP/1.0 200 Ok\r\n";
os << "Content-Type: text/html; charset=\"utf-8\"\r\n";
os << "\r\n";
os << ResponseManager.GetHtml(request);
os << "\n";
tempClientSocket->close();
});
});
//Запуск сервера
if (!tcpServer->listen(QHostAddress::LocalHost, 1488)) {
qDebug() << "Ошибка: порт занят!";
QMessageBox msg;
msg.setText("Сервер не запущен, порт занят!");
msg.exec();
} else {
qDebug() << "Сервер запущен!";
//Запустить интерпретатор веб приложений Microsoft на весь экран

```

```

Sleep(1000);
system("cmd /c start /max mshta http://127.0.0.1:1488/");
}
}
};
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    settings = new QSettings(QDir::currentPath() + "/my_config_file.ini",
    QSettings::IniFormat);
    QCommandLineParser parser;
    QCommandLineOption init("i", "Записать стандартные параметры в ini файл");
    parser.addOption(init);
    parser.process(a);
    if(parser.isSet(init))
    {
        qDebug() << "Инициализация файла настроек...";
        settings->setValue("AppName", "DEFAULTNAME");
        settings->setValue("DRIVER", "QODBC");
        settings->setValue("ConnectionString", "Driver={Microsoft Access Driver
        (*.mdb)};DSN=;DBQ=C:\\Projects\\SmartArchiver\\SmartArchiver.mdb");
        settings->sync();
    }
    //Устанавливаю кодировку QString в UTF8
    QTextCodec::setCodecForLocale(QTextCodec::codecForName("UTF-8"));
    //Подключаюсь к базе данных
    db = QSqlDatabase::addDatabase(settings->value("DRIVER","QODBC").toString());
    db.setDatabaseName(settings->value("ConnectionString","Driver={Microsoft Access
    Driver

```

```

(*.mdb));DSN=";DBQ=C:\\Projects\\SmartArchiver\\SmartArchiver.mdb").toString()
);
if(!db.open())
{
QMessageBox msg;
msg.setText("Не удается подключиться к базе данных!" + db.lastError().text());
msg.exec();
return 0;
};
ConnectionManager manager;
//Закрытие базы данных при завершении
int exec = a.exec();
db.close();
return exec;
}

```