Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

ZIP АРХИВАТОР

БГУИР КР 1-40 02 01 310 ПЗ

Студент:                                                             Кириллов В. И.

Руководитель: Байдун Д. Р.

Минск 2021

**ЗАДАНИЕ**

по курсовому проектированию студента

Кириллова Владимира Игоревича, гр. 050503

1. Тема проекта: «Zip архиватор»

2. Дата выдачи задания: 06.09.2021  
3. Предоставление студентом готового проекта: 15.12.2021

4. Решаемые задачи и функционал разрабатываемого ПО: 4.1. Реализация начального графического интерфейса; 4.2. Реализация функционала работы с директориями и с файлами.

5. ОС, средства разработки и язык программирования: 5.1. Windows 10 x64;  
5.2. Qt Creator; 5.3. C++11;

6. Содержание пояснительной записки:  
Введение. 1. Обзор источников. 2.Обзор методов и алгоритмов решения поставленной задачи. 3.Обоснование выбранных методов и алгоритмов. 4.Описание программы для программиста. 5. Руководство пользователя. Заключение. Список использованных источников. Приложения.

7. Перечень графического материала:  
7.1. Схема структурная. Формат A4. 7.2. Диаграмма классов. Формат A3. 7.3. Схема программы. Формат А4.

Оглавление

[ВВЕДЕНИЕ 5](#_Toc90154972)

[1 Обзор методов и алгоритмов поставленной задачи 6](#_Toc90154973)

[2 Обоснование выбранных методов и алгоритмов 9](#_Toc90154974)

[3 Описание программы для программистов 10](#_Toc90154975)

[4 Описание алгоритмов решения задач 11](#_Toc90154976)

[5 Руководство пользователя 12](#_Toc90154977)

[Заключение 18](#_Toc90154978)

[Литература 19](#_Toc90154979)

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ.............................................................................................................4 1 Обзор источников................................................................................................5 2 Структурное проектирование.............................................................................7 3 Функциональное проектирование......................................................................9 4 Разработка программных модулей...................................................................15 1 Конвертирование ISBN номера из строки в специальный класс..................15 2 Конвертирование ISBN части номера из строки в специальный класс.......16 3 Обновление элемента в базе данных (обработка запроса)............................16 4 Отмена последнего действия (обработка запроса)......................................16

5 Руководство пользователя.................................................................................17 1 Минимальные системные требования..........................................................17 2 Рекомендуемые системные требования........................................................17 3 Комплект поставляемого ПО.........................................................................17 4 Первый запуск программы............................................................................17 5 Как пользоваться программой.......................................................................19 6 «ЧаВо?» (Часто задаваемые вопросы)..........................................................20

6 Тестирование.......................................................................................................21 ЗАКЛЮЧЕНИЕ.....................................................................................................24 ЛИТЕРАТУРА.......................................................................................................25 ПриложениеА........................................................................................................26 Приложение Б........................................................................................................27 Приложение В........................................................................................................28

# ВВЕДЕНИЕ

Язык ассемблера — машинно-ориентированный язык программирования низкого уровня. Представляет собой систему обозначений, используемую для представления в удобно читаемой форме программ, записанных в машинном коде. Его команды прямо соответствуют отдельным командам машины или их последовательностям. Является существенно платформо-зависимым: языки ассемблера для различных аппаратных платформ несовместимы, хотя могут быть в целом подобны.

Язык ассемблера позволяет программисту пользоваться алфавитными мнемоническими кодами операций, по своему усмотрению присваивать символические имена регистрам ЭВМ и памяти, а также задавать удобные для себя схемы адресации (например, индексную или косвенную).

Язык C++ поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование и обобщенное программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе х86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. Важной возможностью языка C++ также является поддержка кода языка ассемблера с использованием блоков \_asm, при помощи чего существует взаимодействие низкоуровневого и высокоуровневого языков программирования.

Таким образом, язык C++ является универсальным и используется в большинстве областей для решения различных задач, поддерживает различные стили программирования и предоставляет программисту максимальную свободу действий, из-за чего широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования.

# 1 Обзор методов и алгоритмов поставленной задачи

Для создания этого программного обеспечения был выбрана технология Qt Creator. Qt Creator (ранее известная под кодовым названием Greenhouse) – кроссплатформенная свободная IDE для разработки на С, C++, JavaScript и QML. Разработана Trolltech (Digia) для работы с фреймворком Qt. Включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML.

Qt Creator позволяет подключать разные библиотеки для соответствующих целей. Для работы с файловой системы были выбраны следующие библиотеки: QDir, QFileDialog, QFileSystemModel, а также класс JLCompress.

* QDir используется для манипулирования путями, доступа к информации о путях и файлах, а также для манипулирования базовой файловой системой. Его также можно использовать для доступа к системе ресурсов Qt.
* QFileDialog предоставляет диалоговое окно, позволяя пользователям выбирать файлы или каталоги‎.
* QFileSystemModel обеспечивает доступ к локальной файловой системе, предоставляя функции для переименования и удаления файлов и каталогов, а также для создания новых каталогов. В простейшем случае его можно использовать с подходящим виджетом отображения в составе браузера или фильтра.
* JLCompress используется для создания zip архивов, сопутствующего сжатия и распаковки существующих zip архивов.

С помощью перечисленных классов можно создавать и распаковывать архивы, также доступен просмотр добавляемых файлов для архивации и выбор файлов с использованием нескольких окон выбора файлов для удобства пользователя. В данном разделе будет рассмотрены основные функциональные составляющие проектируемого ПО.

**Класс Widget**

Рассмотрим основной элемент данного ПО: Систему обновления информации. Данный класс включает в себя большую часть элементов обработки запросов и занимается их обработкой, а также является основным в работе с пользовательским окном. Он реализован с целью взаимодействия конечного пользователя данного программного продукта с системой обработки запросов и обновления данных.

Данный класс содержит поля:

QStringList toArchive – переменная, которая хранит в себе адреса папок с полными именами файлов, которые выбрал пользователь и из которых требуется создать архив.

Ui::Widget \*ui – указатель на собственный класс. Требуется для корректного взаимодействия между методами класса и окнами.

ChooseFilesDialog \*chooseFilesDialog – указатель на класс ChooseFilesDialog. Требуется для вызова окон этого класса и корректного взаимодействия с классом Widget. Сам класс ChooseFilesDialog будет подробно рассмотрен далее.

Методы класса:

void on\_chooseFilesButton\_clicked() – метод, требующийся для пользовательского выбора файлов из окна класса QFileDialog. После выбора файлов помещает их в поле toArchive. Достоинством данного метода является его простота решения поставленной задачи, простое написание программного кода и низкая вычислительная сложность алгоритма.

void on\_CreateArchiveButton\_clicked() – метод, который создает архив из файлов, хранящихся в поле toArchive. В случае если содержимое поля toArchive отсутствует, то вызывается окно класса QMessageBox::critical, в котором пользователю сообщается о соответствующей ошибке. В противном случае вызывается окно QFileDialog::getSaveFileName, в котором пользователь должен выбрать путь создаваемого архива и указать его имя (QString zipArchive). После этого вызывается метод compressFiles класса JLCompress, который сжимает файлы и создает zip архив по переменной zipArchive. В случае успешного создания вызывается окно QMessageBox::information, в котором сообщается, что все прошло успешно. В противном случае вызывается окно QMessageBox::critical, в котором сообщается о соответствующей ошибке. Метод является достаточно простым решением поставленной задачи, достоинствами являются простое написание кода и низкая вычислительная сложность алгоритма. Недостатков не наблюдалось.

void on\_unpackButtonClicked() – метод, который реализовывает распаковку файлов из zip архива. Вызывает окно getOpenFileName класса QFileDialog, где пользователь должен выбрать zip архив в папке. В случае если файл не был выбран вызывается окно QMessageBox::critical, в котором сообщается о соответствующей ошибке и выполнение метода прекращается. В противном случае создается переменная QStringList zipped и вызывается метод JLCompress::getFileList, призванный записать в неё список, содержащий все файлы, находящиеся в архиве. После этого создается переменная folder типа QString , после чего вызывается окно QFileDialog::getExistingDirectory, который записывает выбранный пользователем в окне путь к папке для распаковки файлов. Далее вызывается метод extractFiles класса JLCompress, который производит распаковку файлов. Так как метод extractFiles имеет тип возвращаемого значения bool, производится проверка на корректность распаковки: в случае если распаковка произведена успешно, вызывается окно QMessageBox::information, в котором сообщается об удачной распаковке, и QMessageBox::critical в случае если распаковка прошла неуспешно. Метод является достаточно простым решением поставленной задачи, достоинствами являются простое написание кода и низкая вычислительная сложность алгоритма. Недостатков не наблюдалось.

void on\_clearButton\_clicked() – метод, который очищает поле файлов для архивации toArchive и удаляет соответствующие файлы из списка в графическом интерфейсе.

void copyFiles(QStringList &files) – метод, который добавляет файлы из files в поле toArchive и отображает их через listView.

void on\_chooseFilesButtonAlter\_clicked() – метод, реализовывающий работу с несколькими окнами выбора файлов для архивации самим пользователем. Вызывает создание нового окна chooseFilesDialog класса ChooseFilesDialog, после чего при помощи метода show() вызывается отображение данного окна. Подробно работа методов этого класса будет рассмотрена далее. После выбора файлов при помощи функции connect, связывающей классы Widget и ChooseFilesDialog, список файлов для архивации будут отправлены в класс Widget полю toArchive, рассмотренному подробно выше. Таким образом, по нажатию на кнопку chooseFilesButtonAlter, можно вызывать множество окон класса ChooseFilesDialog, что обеспечивает большую удобность в работе с программой.

**Класс ChooseFileDialog**

Как упоминалось выше, данный класс реализует работу пользователя с несколькими окнами для выбора файлов для архивации. Он реализован с целью взаимодействия конечного пользователя данного программного продукта с системой обработки запросов и обновления данных. Данный класс содержит данные поля:

QFileSystemModel \*fileSystemModel – указатель, с помощью которого производится работа с файловой системой.

Ui::MainWindow \*ui – указатель на собственный класс. Требуется для корректного взаимодействия между методами класса и окнами.

QStringList files – поле, являющееся списком полных путей файлов для архивации.

Методы класса:

void on\_listView\_doubleClicked(const QModeindex &index) – метод, который отслеживает двойное нажатие мыши пользователя по области. При вызове создает объект listView класса QListView, требующийся для графического отображения файлов и объект fileInfo класса QFileInfo, требующийся для обработки информации о файлах. Далее осуществляется переход между папками. Присутствуют две ячейки списка, предназначенные для перехода в корневую папку программы и в папку выше. В графическом интерфейсе они обозначены «.» и «..» соответственно. Далее в методе следует проверка, является ли выбранный пользователем объект папкой или файлом. В первом случае с помощью метода setRootIndex(index) объекта listView делаем переход в следующую папку, а во втором с помощью метода QStringList::push\_back(fileInfo.absoluteFilePath()) добавляем в список файлов для архивации выбранный пользователем файл. Недостатком является невозможность смены выбранного ранее файлового диска, однако в остальном проблем не обнаружено и метод является достаточно простым решением поставленной задачи. Достоинствами являются простое написание кода и низкая вычислительная сложность алгоритма.

void on\_ok\_cancel\_button\_accepted() – метод, который обрабатывает пользовательское нажатие кнопки OK. При вызове эмитирует сигнал mainWindow\_ChooseFiles(files), после чего закрывает окно выбора файлов. Сигнал mainWindow\_ChooseFiles(QStringList &files) требуется для передачи файлов из класса ChooseFilesDialog в класс Widget.

void on\_ok\_cancel\_button\_rejected() – метод, который обрабатывает пользовательское нажатие кнопки Cancel и просто закрывает окно выбора файлов.

# 2 Обоснование выбранных методов и алгоритмов

Метод on\_listView\_doubleClicked(const QModelIndex &index) класса ChooseFilesDialog является одним из основных методов данного программного обеспечения. Он отображает все папки и файлы устройства пользователя и дает возможность перехода между папками и выбора файлов для архивации. Данный метод является достаточно простым в написании программного кода. Вычислительная сложность алгоритма перехода между папками и открытие файлов достаточно низка – O(1), так как при вызове метода требуется обработать index, и в зависимости от этого либо добавлять файл в список для архивации, либо перейти в другую папку. Недостатком данного метода является невозможность смены выбранного файлового диска после однократного выбора.

Метод on\_createArchiveButton\_clicked() класса Widget является одним из основных методов данного программного обеспечения. Он используется для сжатия и архивации файлов в архив. Данный метод является достаточно простым в написании программного кода. Вычислительная сложность алгоритма – O(n), так как при вызове метода требуется обработать и сжать количество файлов, равное toArchive.count().

Метод on\_unpackButton\_clicked() класса Widget является одним из основных методов данного программного обеспечения. Он используется для разархивации файлов из архива. Данный метод является достаточно простым в написании программного кода. Вычислительная сложность алгоритма – O(2n), так как при вызове метода требуется узнать количество файлов в архиве, после чего для каждого провести операцию извлечения.

# 3 Описание программы для программистов

По ходу работы придерживались правила паттернов Фасад (Facade) и Одиночка (Singleton). Singleton – это порождающий паттерн, который гарантирует существование только одного объекта определённого класса, а также позволяет достучаться до этого объекта из любого места программы. Это можно увидеть в файле main.cpp, где осуществляется создание единственного экземпляра класса Widget, что является главным принципом паттерна Singleton.

Также в работе использовался паттерн Facade. Суть паттерна заключается в предоставлении простого интерфейса к более сложной системе классов, библиотеке или фреймворку. Паттерн «Фасад» предоставляет унифицированный интерфейс вместо набора интерфейсов некоторой подсистемы. Фасад определяет интерфейс более высокого уровня, который упрощает использование подсистемы. Проще говоря, «Фасад» — есть ни что иное, как некоторый объект, аккумулирующий в себе высокоуровневый набор операций для работы с некоторой сложной подсистемой. Фасад агрегирует классы, реализующие функциональность этой подсистемы, но не скрывает их. Важно понимать, что клиент, при этом, не лишается более низкоуровневого доступа к классам подсистемы, если ему это, конечно, необходимо. Фасад упрощает выполнение некоторых операций с подсистемой, но не навязывает клиенту свое использование.

Использование данного паттерна можно увидеть в файле main.cpp, где и находится точка входа в программу. Также классы ChooseFilesDialog и Widget связаны между собой отношением ассоциации, а точнее класс ChooseFilesDialog является композицией для класса Widget.

В данном программном обеспечении роль интерфейса и глобальной точки входа в программу является класс Widget. Класс ChooseFilesDialog является еще одним классом, созданным по принципу паттерна проектирования Facade и Singleton, так как позволяет создавать множество окон данного класса.

Таким образом, использование основных принципов паттернов проектирования Singleton и Facade позволило создать удобный и простой пользовательский интерфейс, который будет прост для пользователя.

Схема 1. Структура файлов

# 4 Описание алгоритмов решения задач

Алгоритм выбора файлов с использованием метода Widget::on\_chooseFilesButton\_clicked():

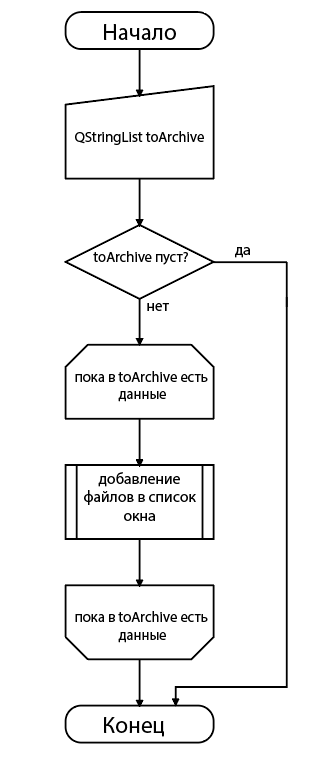


Схема 2. Алгоритм выбора файлов

Алгоритм выбора файлов с использованием метода Widget::on\_chooseFilesButtonAlter\_clicked():

Шаг 1. Создание окна класса ChooseFilesDialog для выбора файлов;

Шаг 2. При двойном щелчке ЛКМ происходит вызов метода ChooseFileDialog::on\_listView\_doubleClicked(const QModelIndex &index), обрабатывающего действия пользователя;

Шаг 3. Если двойной щелчок ЛКМ был по папке, то происходит переход в нее;

Шаг 4. Если двойной щелчок ЛКМ был по файлу, то тот добавляется в список файлов для архивации;

Шаг 5. Если двойной щелчок ЛКМ был по полю «..», то происходит переход в корневую папку диска;Шаг 6. Если двойной щелчок ЛКМ был по полю «…», то происходит переход в папку расположения программы;

Шаг 7. В случае нажатия кнопки OK происходит имитация сигнала ChooseFilesDialog::mainWindow\_ChooseFiles(files) (см. пункт 8, иначе см. шаг 10);

Шаг 8. Сигнал поступает в слот Widget::copyFiles(QStringList &files)

Шаг 9. Файлы из списка QStringList files записываются в поле; toArchive класса Widget, а также отображаются в графическом интерфейсе объекта listWidget;

Шаг 10. Происходит закрытие окна ChooseFilesDialog;

Шаг 11. Завершение работы алгоритма.

Алгоритм создания архива с использованием метода Widget::on\_CreateArchiveButton\_clicked():

Шаг 1. Удаление дубликатов поля toArchive методом QStringList::removeDuplicates() во избежание ошибок при архивации;

Шаг 2. Определение количества файлов для архивации;

Шаг 3. Проверяется количество файлов (в случае если оно равно 0, см. пункт 4, в противном случае см. пункт 8);

Шаг 4. Создание окна метода QFileDialog::getSaveFileName(), в котором пользователь выбирает файлы для архивации;

Шаг 5. Сжатие выбранных файлов;

Шаг 6. Проверка результата сжатия (если успешно, см. пункт 8, в противном случае см. пункт 7);

Шаг 7. Вызов окна метода QMessageBox::critical() с сообщением о соответствующей ошибке;

Шаг 8. Завершение работы алгоритма.

Алгоритм обработки двойных кликов ЛКМ по выбранной папке или файлу с использованием метода

void on\_listView\_doubleClicked(const QModeindex &index):

Шаг 1. Создание объектов listView класса QListView и fileInfo класса QFileInfo;

Шаг 2. Проверка на тип объекта, на который нажал пользователь;

Шаг 3. Если выбор пал на папку, производится переход в неё посредством метода QListView::setRootIndex(index);

Шаг 4. Если выбор пал на файл, то тот добавляется в архив с помощью метода QStringList::push\_back();

Шаг 5. Если выбор пал на кнопку перехода в корневую папку программы, производится переход туда с помощью метода QDir::cdUp();

Шаг 6. Если выбор пал на кнопку перехода в родительскую папку, производится переход в нее при помощи метода QListView::setRootIndex(QFileSystemModel filesystemmodel->index(“/”));

Шаг 7. Если в меню пользователь нажал кнопку OK, происходит имитация сигнала, и выбранные файлы становятся доступными для архивации, см. пункт 9. Если же была нажата кнопка Cancel (отмена), см. пункт 9;

Шаг 9. Завершение работы алгоритма.

# 5 Руководство пользователя

В данном разделе приведена вся информация, которая должна публиковаться вместе с продуктом.

1. **Минимальные системные требования**

* ОС: Windows XP – 10 x64
* ОЗУ: 128 МБ
* ПЗУ: 70 МБ
* Мышь

1. **Рекомендуемые системные требования**

* OC: Windows XP – 10 x64
* ОЗУ: 256 МБ
* ПЗУ: 100 МБ
* Мышь

1. **Комплект поставляемого программного обеспечения**

В комплекте с поставляемым программным обеспечением (далее - ПО) должны быть: исполняемый файл “ZipArchiver.exe”, библиотеки quazip.dll и zlib.dll.

1. **Первый запуск программы**

Для запуска программы требуется:

1. Проверить комплект поставляемого ПО;
2. Минимальные системные требования;
3. Запустить файл “ ZipArchiver.exe ”(см. рисунок 5.1);

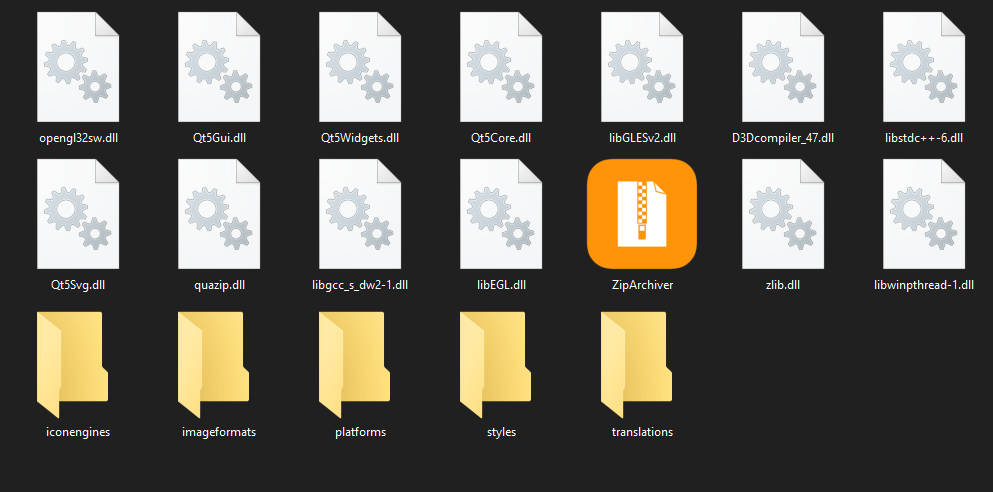


Рисунок 5.1 – папка с программой

Далее будет рассмотрен представлен снимок экрана базового окна приложения с пояснением каждой функции:

1. Кнопка для выбора файлов
2. Кнопка для выбора файлов с возможностью использования нескольких окон
3. Кнопка для создания архива
4. Кнопка для очистки файлов из списка (6)
5. Кнопка для разархивации файлов
6. Список файлов для архивации

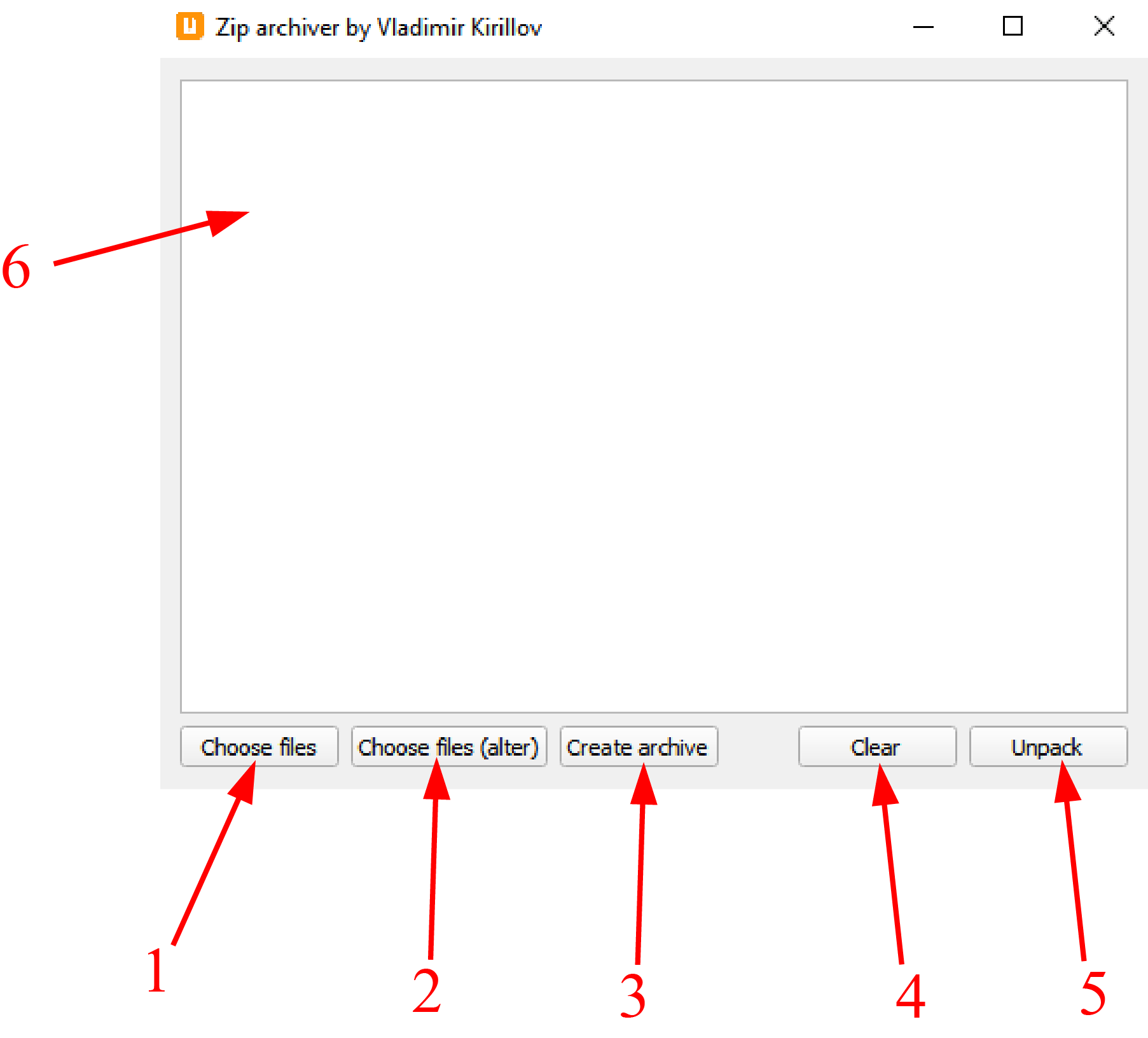


Рисунок 5.2 – первый запуск программы

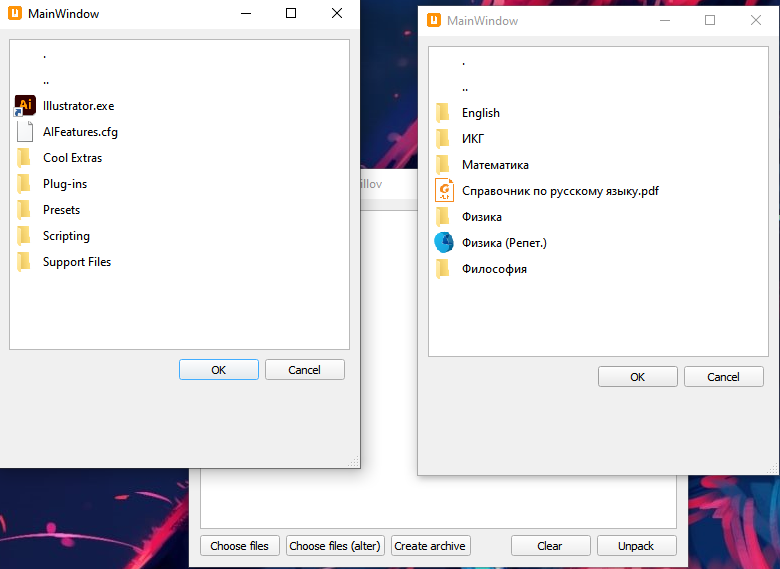


Рисунок 5.3 – переход между папками в нескольких окнах

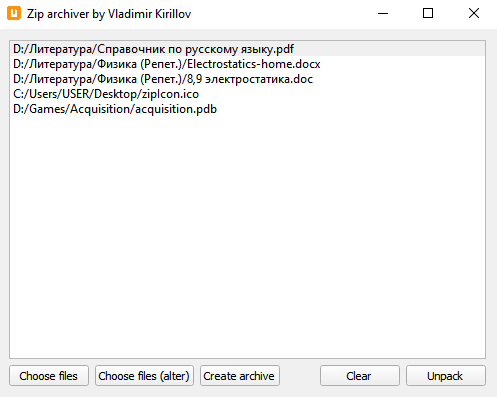


Рисунок 5.4 – меню выбора операции со списком файлов для архивации

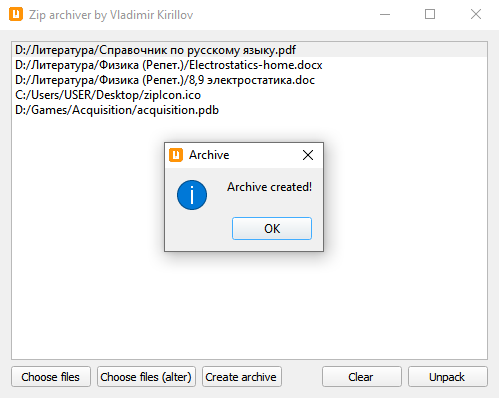


Рисунок 5.5 – меню создание архива

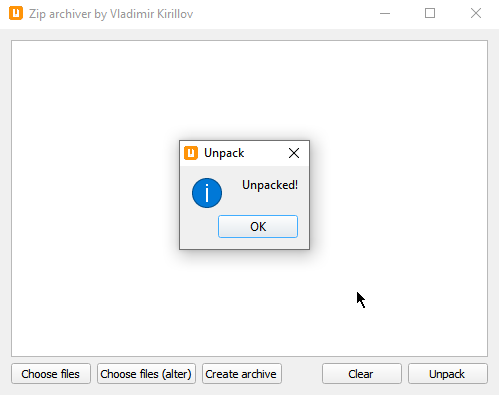


Рисунок 5.6 – меню, сообщающее об успешной распаковке архива

На данных рисунках изображены все пункты меню и основные возможности данной программы: создание и распаковка zip архивов.

На рисунке 5.3 изображены меню для перехода в разные папки. В меню отображаются как файлы, так и папки. При двойном нажатии ЛКМ по папке производится переход в нее, а по файлу – отправление его в список для архивации.

На рисунке 5.4 изображено меню операций с заполненным списком файлов для архивации. На данном этапе возможна операция архивации.

На рисунках 5.5 и 5.6 изображены меню, сообщающие пользователю об успешной архивации и разархивации файлов.

# Заключение

В ходе разработки данного приложения были получены знания по проектированию приложений с графическим интерфейсом, обработчиком событий, файловой системой, а также получено более глубокое понимание концепции объектно-ориентированного программирования и знания по работе фреймворка Qt.

Также были приобретены знания по работе с событиями, сигналами и слотами, по имитации сигналов, по связыванию графической и логической частей приложения, а также закреплены знания по работе с файлами.

Из достоинств приложения стоит выделить быстроту работы, хороший показатель сжатия (до 85%), возможность распаковывать файлы из любого подходящего zip архива, а также очень низкие требования для работы на всех версиях Windows XP – 10, а также достаточно простой для рядового пользователя дизайн.

Из недостатков можно выделить необходимость наличия 64-разрядной версии Windows, а также невозможность работы с другими форматами архивов типа .rar, .tar и т.д. Также стоит отметить тот факт, что приложение разрабатывалась по принципам Singleton и Facade, что негативно влияет на модульность приложения.

В дальнейшем данный продукт может получить ряд важных дополнительных функций. Помимо возможности работы с вышеописанными форматами архивов, возможно интегрировать его в файловый менеджер, сделав архиватор встроенным, а также применить другой подход к проектированию, к примеру MVC, таким образом сделав приложение модульным и более простым к поддержке и улучшению.

Подводя итог, данный продукт получил хорошее развитие в процессе написания и лабораторных работ по смежным темам, что позволило интегрировать некоторые знания и принципы. Также данный проект помог понять структуру и процесс разработки приложений с графическим интерфейсом, показать их достоинства и недостатки, а также показал дальнейшие пути повышения качества приложений.

# Литература

1. Документация фреймворка Qt. – [Электронный ресурс]. – Адрес ресурса: <https://doc.qt.io> – Дата доступа 10.10.2021
2. Официальный сайт фреймворка Qt. – [Электронный ресурс]. – Адрес ресурса: <https://www.qt.io> – Дата доступа 10.10.2021
3. Шилдт Герберт, C++: базовый курс, 3-е изд. : Пер. с англ. – М. : ООО «И.Д. Вильямс», 2017, 624 с.
4. Паттерны проектирования. – [Электронный ресурс]. – Адрес ресурса: <https://refactoring.guru/ru/design-patterns> – Дата доступа: 08.11.2021
5. Луцик Ю.А. Объектно-ориентированное программирование на языке С++ / Ю.А.Луцик, А.М.Ковальчук, И.В.Лукьянова. – Мн.: БГУИР, 2003
6. Интернет-уроки по C++. - [Электронный ресурс]. - Адрес ресурса: <https://ravesli.com/uroki-cpp> - Дата доступа 10.09.2021.