

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

Приложение «Музыкальный плеер»
по дисциплине

«Конструирование программ и языки программирования»

БГУИР КП 1–40 02 01 313 ПЗ

Студент

Липский Г.В.

Руководитель

Байдун Д.Р.

Минск 2021

ЗАДАНИЕ

для курсового проектирования

студента Липского Григория Васильевича, гр. 050503

1. Тема проекта: «Музыкальный плеер»
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 графического интерфейса;
 - 4.2. Реализация функционала для взаимодействия пользователя с приложением.
5. ОС, средства разработки и язык программирования:
 - 5.1. Windows 10 Pro;
 - 5.2. QT Creator;
 - 5.3. C++.
6. Содержание пояснительной записки:

Введение. 1. Обзор источников. 2. Структурное проектирование.

3. Функциональное проектирование. 4. Разработка программных модулей.

5. Тестирование. 6. Руководство пользователя. Заключение. Список литературы. Приложения.
7. Перечень графического материала:
 - 7.1. Диаграмма классов. Формат А4;
 - 7.2. Блок-схема алгоритма. Формат А4.

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов курсовой работы	Объём этапа, %	Срок выполнения этапа	Примечания
Оформление плана курсового проекта	15	06.09 - 15.10	
Разработка программного обеспечения	50	15.10 - 15.11	
Оформление пояснительной записки	35	15.11 - 15.12	С выполнением чертежей

РУКОВОДИТЕЛЬ

(подпись)

(ФИО)

Задание принял к исполнению

(подпись)

(ФИО)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
ОБЗОР ИСТОЧНИКОВ.....	5
СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ.....	8
ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ.....	10
РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ.....	12
ТЕСТИРОВАНИЕ.....	15
РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	19
ЗАКЛЮЧЕНИЕ.....	23
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	24
ПРИЛОЖЕНИЕ А.....	25
ПРИЛОЖЕНИЕ Б.....	27
ПРИЛОЖЕНИЕ В.....	29

ВВЕДЕНИЕ

Музыка помогает человеку отвлечься от житейских забот и хлопот, а также настроиться на определённый событийный ряд. С её помощью можно расслабиться. Музыка выступает в качестве эффективного способа погружения в определённое состояние. Она выступает источником эмоций. Ввиду личной заинтересованности для данного курсового проекта в качестве программы для разработки было выбрано приложение «Музыкальный плеер». Помимо практического интереса, тема имеет широкие возможности для последующей модернизации проекта с применением навыков, полученных в ходе изучения курса КПиЯП.

Объектно-ориентированное программирование представляет собой технологию программирования, которая базируется на классификации и абстракции объектов. Одним из наиболее популярных средств объектно-ориентированного программирования, позволяющим разрабатывать программы, эффективные по объёму кода и скорости выполнения является C++.

C++ — компилируемый, статически типизируемый язык общего назначения. Он поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое 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 в себе как возможности низкоуровневых языков программирования, так и возможности высокоуровневых.

Основными концепциями объектно-ориентированного программирования являются инкапсуляция, полиморфизм и наследование. Язык C++ предоставляет исчерпывающие возможности для реализаций этих концепций. В результате использования инкапсуляции, программа,

написанные на C++, обладает повышенной защищённостью объектов от влияния на них кода других частей этой же программы. Наследование предоставляет важную возможность повторного использования кода, что может значительно уменьшить количество кода, однако требует предварительного проектирования архитектуры. Полиморфизм позволяет программе быть более гибкой. Такая система удобна в тестировании и позволяет легко заменять и модифицировать свои компоненты.

Универсальность и гибкость языка позволяют использовать его в различных целях. Область применения данного языка включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений. Исходя из этого можно считать, что данный язык достаточно удобен для написания выбранной курсовой работы.

Данная работа посвящена созданию MP3 плеера с функциями обычного проигрывателя.

Целью данной работы является разработка MP3 плеера.

Задачи данного проекта:

1. Описать и разработать MP3 проигрыватель;
2. Создать удобный пользовательский интерфейс;
3. Создать регулятор громкости;
4. Создать строку описания треков;
5. Разработать стандартный функционал проигрывателя.

1 ОБЗОР ИСТОЧНИКОВ

1.1 Анализ аналогов программного средства

В наше время существует множество различных популярных аудиопроигрывателей: AIMP, iTunes, Winamp, JetAudio, XMplay, KMPlayer и другие.

У всех них главной задачей является воспроизведение аудиофайлов разного формата. Однако все вышеперечисленные приложения включают в себя различный функционал и возможности. Очевидно, что чем выше качество продукта, тем большее количество людей будет им пользоваться. У всех проигрывателей есть свой уникальный интерфейс. Интерфейс играет важную роль при выборе проигрывателя для повседневных целей. Если приложение будет обладать обширным функционалом, но не будет выглядеть приятно для глаз, часть пользователей перестанет пользоваться этим приложением и выберет какой-нибудь похожий аналог. Функционал проигрывателя играет более весомую роль при выборе приложения пользователем. Все вышеперечисленные приложения включают в себя обширный функционал и приятный интерфейс. Поэтому выбор большинства рядовых пользователей падает именно на них.

Области применения мультимедиа, в том числе: локализация трехмерного звука, сетевые игры для большого числа участников, поддержка нового периферийного оборудования и устройства ввода данных, видео, связи с различными источниками информации.

Специальными исследованиями установлено, что из услышанного в памяти остается только четверть, из увиденного — треть, при комбинированном воздействии зрения и слуха — 50%, а если вовлечь учащегося в активные действия в процессе изучения при помощи мультимедийных приложений — 75%.

Областью применения MP3 проигрывателя является частное

прослушивание MP3 файлов на своем ПК.

Проигрыватель можно использовать для воспроизведения мультимедийных файлов с расширением MP3, находящихся на компьютере или другом внешнем накопителе.

Формат MP3 (более точно, англ. MPEG-1/2/2.5 Layer 3; но не MPEG-3) – третий слой формата кодирования звуковой дорожки MPEG, лицензируемый формат файла для хранения аудиоинформации.

MP3 является одним из самых распространённых и популярных форматов цифрового кодирования звуковой информации с потерями. Он широко используется в файлообменных сетях для оценочной передачи музыкальных произведений. Формат может проигрываться практически во всех популярных операционных системах, на большинстве портативных аудиоплееров, а также поддерживается всеми современными моделями музыкальных центров и DVD-плееров.

1.2 Постановка задачи

Программа должна иметь понятный пользователю интерфейс и включать в себя основные функции музыкального проигрывателя:

- Добавление одного или нескольких аудиофайлов в плейлист. Будет реализован множественный выбор файлов: пользователь, нажимая на кнопку добавления аудиофайлов в плейлист, сможет выбрать один или несколько аудиофайлов сразу.
- Воспроизведение композиций. Если плейлист не будет пустым на момент нажатия кнопки воспроизведения, будет воспроизводиться выбранная пользователем композиция. В случае если плеер находится “в состоянии паузы”, при нажатии на кнопку будет продолжено воспроизведение текущей композиции.
- Установка на паузу. У пользователя будет возможность временно

приостановить воспроизведение аудиофайла для дальнейшего продолжения прослушивания.

- Навигация по аудиофайлу. Перемещение вперёд-назад по аудиофайлу на некоторое количество секунд.

- Регулирование звука. Будет реализована возможность одним кликом выключить звук воспроизведения, а также регулирование звука с помощью ползунка.

- Переход к следующей или предыдущей композиции.

- Вывод информации о текущей композиции, а именно название трека и его длина.

2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

2.1 Сторонние программные компоненты

Для упрощения разработки будет использоваться QT Creator – кроссплатформенная свободная IDE для разработки на C, C++, JavaScript и QML. Основная задача Qt Creator – упростить разработку приложения с помощью фреймворка Qt на разных платформах.

В Qt Creator реализовано автодополнение, в том числе ключевых слов, введённых в стандарте C++11, подсветка кода. Также есть возможность задания стиля выравнивания, отступов и постановки скобок.

Был подключён модуль multimedia (QT += core gui multimedia) для доступа к классам QMediaPlayer и QMediaPlaylist, на основе которых реализован аудиоплеер.

Для реализации отображения плейлиста, использован QStandardItemModel. В него помещены пути к аудиофайлам, а также их названия. В первой колонке будет название аудиофайла, а во второй будет полный путь, но данная колонка будет скрыта в объекте QTableView, который будет отвечать за отображение плейлиста.

Для реализации интерфейса был подключён класс QWidget.

2.2 Структура приложения

В приложении можно выделить несколько основных элементов: интерфейс, блок взаимодействия пользователя с аудиоплеером, блок плейлиста.

Интерфейс представляет из себя окно, в котором располагаются блоки взаимодействия и плейлиста. То есть можно сказать, что интерфейс связан с блоком взаимодействия пользователя и с блоком плейлиста.

Блок плейлиста отвечает за отображение композиций, загруженных в

программу.

Блок взаимодействия пользователя с аудиоплеером отвечает за воспроизведение аудиофайлов и за другие действия, которые пользователь может запросить. Выполнение этих действий лежит в методах `on_vol_clicked()`, `on_right_clicked()`, `on_btn_add_clicked()`, `on_btn_play_clicked()`, `on_left_clicked()` пользовательского класса `Widget`, унаследованного от библиотечного класса `QWidget`. А также библиотечных классов `QApplication`, `QToolButton`, `QSlider`, `QLabel`, `QTableView`.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1 Описание функционирования программы

Работа приложения основана на взаимодействии класса `Widget`, его методов и класса `QWidget` из библиотеки с таким же названием, а также объекта `widget` класса `Widget`.

За добавление аудиофайла в плейлист отвечает метод класса `Widget` `on_btn_add_clicked`. В этом методе также предусмотрен множественный выбор файлов, т.е. одним вызовом метода пользователь имеет возможность добавить два и более аудиофайла в плейлист.

После добавления одного или нескольких аудиофайлов в плейлист пользователь при помощи кнопки `QToolButton btn_play`, связанной с методом `on_btn_play_clicked` класса `Widget` имеет возможность воспроизвести выбранный аудиофайл. Аудиофайл будет воспроизводиться до того момента, пока пользователь не нажмёт на кнопку `QToolButton btn_pause`, в результате чего аудиофайл воспроизводиться не будет до того момента, пока пользователь снова не вызовет метод `on_btn_play_clicked`.

Помимо этого пользователь может перематывать аудиофайл вперёд-назад, с помощью кнопок `QToolButton left` и `QToolButton right` соответственно, которые вызывают методы `on_left_clicked` и `on_right_clicked`.

Метод `on_vol_clicked` отвечает за регулирование громкости воспроизведения аудиофайла. Нажатием кнопки `QToolButton vol` пользователь может включить или выключить звук в зависимости от текущего значения громкости. Помимо использования кнопки, пользователь имеет возможность устанавливать громкость воспроизведения с помощью `QSlider volume`.

Выбирать аудиофайл для воспроизведения пользователь может в `QTableView playlistView`. Название текущего аудиофайла, а также его длина в формате “минуты:секунды” отображается в `QLabel cur_c` и `QLabel label_2` соответственно.

3.2 Описание основных функций взаимодействия программы с пользователем

Главный класс Widget управляет всем процессом работы аудиоплеера и находится в namespace Ui. Он содержит в себе девять полей секции private, два поля секции public.

Поля public-секции explicit Widget(QWidget *parent = 0) и ~Widget() являются конструктором и деструктором класса.

Поле Ui::Widget *ui является указателем на объект класса.

Поля private-секции QStandardItemModel *m_playListModel, QMediaPlayer *m_player и QMediaPlaylist *m_playlist являются моделями данных для отображения плейлиста, проигрывания треков и плейлиста проигрывателя соответственно.

Методы on_vol_clicked(), void on_rght_clicked(), void on_btn_add_clicked(), void on_btn_play_clicked() и void on_left_clicked() являются главными методами функционирования приложения. Они служат для регулирования громкости, перемотки аудиофайла вперёд, добавления одного или нескольких аудиофайлов в плейлист, воспроизведения аудиофайла и перемотки аудиофайла назад соответственно.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Схема алгоритма функции void Ui::on_btn_add_clicked()

Схема алгоритма метода представлена на чертеже ГУИР.400201.313 Д.1.

Данный метод предназначен для добавления одной или нескольких композиций в плейлист для дальнейшего воспроизведения.

4.2 Алгоритм по шагам функции void Ui::on_right_clicked()

Данная функция служит для перемотки аудиофайла на 10 секунд вперёд.

1. Начало.
2. Получаем длину текущего аудиофайла: `dur = m_player->duration()`.
3. Получаем текущую позицию в аудиофайле: `pos = m_player->position()`.
4. Сдвигаем позицию на 10000 миллисекунд вперед: `pos = pos + 10000`.
5. Если `pos < dur`, то шаг 6, иначе шаг 7.
6. Устанавливаем новую позицию: `m_player->setPosition(pos)`.
7. Автоматическое нажатие кнопки `btn_next`: `ui->btn_next->click()`.
8. Конец.

4.3 Алгоритм по шагам функции void Ui::on_left_clicked()

Данная функция служит для перемотки аудиофайла на 10 секунд назад.

1. Начало.
2. Получаем текущую позицию в аудиофайле: `pos = m_player->position()`.
3. Сдвигаем позицию на 10000 миллисекунд назад: `pos = pos - 10000`.
4. Если `pos <= 0`, то шаг 5, иначе шаг 6.
5. Устанавливаем аудиофайл в нулевую позицию: `m_player->setPosition(0)`.
6. Устанавливаем новую позицию: `m_player->setPosition(pos)`.

7. Конец.

4.4 Алгоритм по шагам функции void Ui::on_vol_clicked()

Данная функция служит для регулирования громкости воспроизведения.

1. Начало.
2. Если `ui->volume->sliderPosition() != 0`, то шаги 3-5, иначе шаги 6-8.
3. Устанавливаем громкость, равную 0: `ui->volume->setValue(0)`.
4. Устанавливаем слайдер volume в нулевую позицию (минимальная позиция):
`ui->volume->setSliderPosition(0);`
5. Устанавливаем иконку кнопки:
`ui->vol->setIcon(QIcon(QPixmap(":/buttons/zvuk_of.png")));`
6. Устанавливаем громкость, равную 100: `ui->volume->setValue(100);`
7. Устанавливаем слайдер volume в сотую позицию (максимальная позиция):
`ui->volume->setSliderPosition(100);`
8. Устанавливаем иконку кнопки:
`ui->vol->setIcon(QIcon(QPixmap(":/buttons/zvuk_on.png")));`
9. Конец

4.5 Алгоритм по шагам функции void Ui::on_btn_play_clicked()

Данная функция служит для воспроизведения аудиофайла.

1. Начало
2. Получаем длину текущего аудиофайла: `dur = m_player->duration();`
3. Вычисляем количество полных минут в текущем аудиофайле:
`m = dur/1000/60;`
4. Вычисляем количество полных секунд в текущем аудиофайле:
`s = dur/1000%60;`
5. Если `s > 9`, то шаг 6, иначе шаг 7.
6. В `label_2` выводим продолжительность аудиофайла в формате

“минуты:секунды”: ui->label_2->setText(QString("%1%2%3%4").arg("Длина трека: ").arg(m).arg(":").arg(s));

7. В label_2 выводим продолжительность аудиофайла в формате “минуты:секунды” с добавлением нуля перед количеством секунд:
ui->label_2->setText(QString("%1%2%3%4").arg("Длина трека:").arg(m).arg(":0").arg(s));

8. Конец.

5 ТЕСТИРОВАНИЕ

Контрольный пример предназначен для оценки правильности работы программы. В данном случае, контрольный пример предназначен для оценки корректности использования проигрывателя и его компонентов. Исходными данными являются файлы MP3.

В результате проверки выяснилось, что контрольные результаты совпадают с полученными. Вывод: программа работает правильно.

Тестирование является одним из важнейших этапов жизненного цикла, направленным на повышение качественных характеристик. Качество программного средства очень сильно зависит от того, насколько хорошо он выполняет задачи, для которых был создан. Скрытые ошибки могут сильно изменить результат работы программного продукта и тем самым привести к ошибочному выполнению поставленных задач. Естественно, что такие проявления отрицательно сказываются не только на работе, но и на отношении пользователей к продукту, имеющему ошибки, поэтому практически все разработчики программного обеспечения (включая корпорации-гиганты Microsoft, Sun, IBM и т. д.) очень серьезно относятся к процессу тестирования работоспособности своих продуктов.

При создании типичного программного продукта около 40% общего времени и более 40% общей стоимости расходуется на проверку разрабатываемой программы.

Тестирование - это процесс многократного выполнения программы с целью обнаружения ошибок.

Процесс тестирования должен проводиться не автором программы (модуля). Этот принцип был выбран из тех соображений, что процесс тестирования это процесс деструктивный. Следовательно, он будет особенно труден и малоэффективен для автора модуля, так как после выполнения конструктивной части при проектировании и написании программы ему будет сложно перестроиться на деструктивный образ мышления и, создав программу

(модуль), тут же приступить к пристрастному выявлению ошибок. Это объясняется психологическими особенностями мышления человека. Описание предполагаемых значений должно быть необходимой частью тестового набора данных.

Необходимо наиболее тщательно тестировать те модули, в которых обнаружено наибольшее количество ошибок.

Кроме правильных тестовых последовательностей должны присутствовать и неправильные (непредусмотренные) последовательности.

Необходимо проверять не только то, что делает программа из того, для чего она предназначена, но и не делает ли она то, что не должна делать.

Тестирование программ является одной из составных частей общего понятия - "отладка программ". Если тестирование – это процесс, направленный на выявление ошибок, то целью отладки являются локализация и исправление выявленных в процессе тестирования ошибок.

При проведении тестирования встает вопрос о том, когда завершить тестирование, когда разрабатываемое ПС достигло того уровня надежности, которое может удовлетворить будущих пользователей.

В процессе разработки программного продукта тестирование проводилось на разных этапах разработки. Первое тестирование – на этапе проектирования.

При этом тестировании выяснялось, соответствует ли проект требованиям к программному продукту, указанным в техническом задании, учитываются ли все условия работы и все данные, соответствуют ли требования проекта аппаратным и программным ресурсам заказчика, правильно ли выбраны инструментальные средства разработки.

Следующее тестирование проводилось на стадии кодирования. Здесь использовались статические методы тестирования. Из статистических методов использовался метод инспекции исходного текста. При просмотре текста программы ошибок не было обнаружено.

Для статического тестирования программного кода среда разработки QT Creator предлагает автоматическую проверку синтаксиса языка. При обнаружении синтаксической ошибки, выдается сообщение о ней, указывается причина ошибки, и курсор ставится в то место программы, где обнаружена ошибка. Вообще, возникающие ошибки можно разделить на такие три вида:

1. ошибки компиляции;
2. ошибки периода выполнения;
3. логические ошибки.

Ошибки компиляции проистекают из ошибок в тексте кода. Также они включают в себя ошибки в синтаксисе.

Логические ошибки имеют место, когда приложение работает не так, как это планировалось разработчиком. Приложение может иметь синтаксически правильный код, не выполнять недопустимых операций, но, однако, выдавать неправильные результаты. Обнаружить логические ошибки можно только в ходе тестирования и анализа результатов.

Было проведено тестирование программы на выявление ее поведения при различных проблемах, связанных с доступом к базам данных. Для этого специально были созданы ситуации, когда базы данных отсутствовали, либо указывали на неправильные источники данных.

В ходе тестирования программы был проведен анализ качества разработанного программного продукта, на основании которого можно утверждать, что данная система прошла контроль системы показателей качества и может быть использована по назначению.

На этапе тестирования проверяется правильность выполнения основных действий, совершаемых в ходе работы плеера.

Ниже приведены скриншоты выполнения основных функций приложения.

Добавление аудиофайла в плейлист (рис. 5.1).

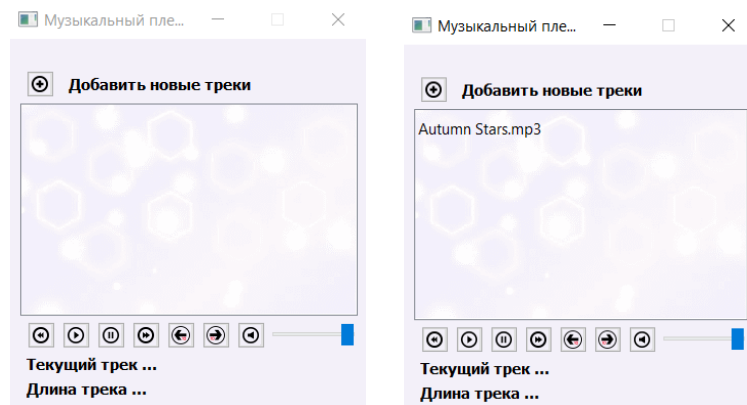


Рисунок 5.1

Воспроизведение композиции (рис. 5.2).

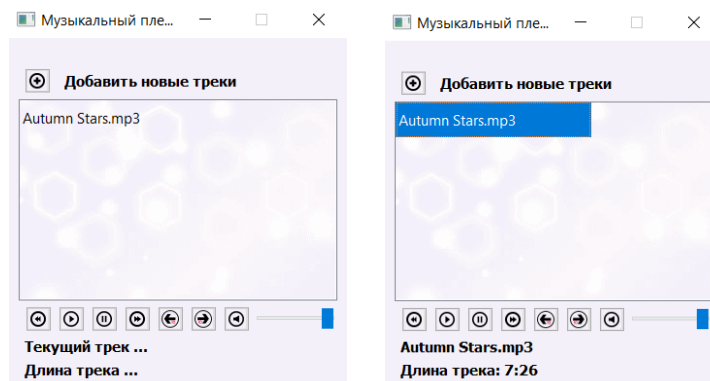


Рисунок 5.2

Выключение звука (рис. 5.3).

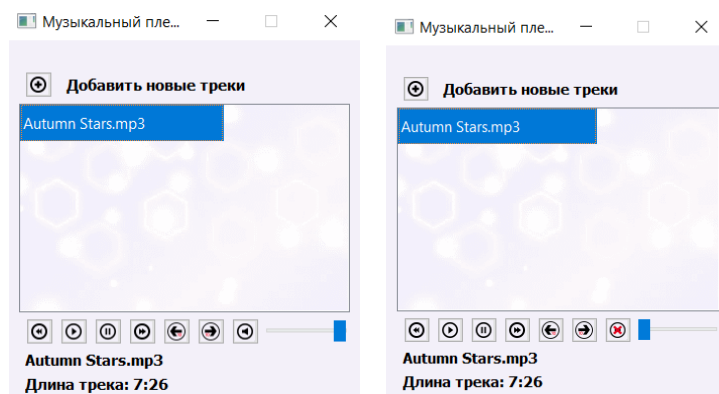


Рисунок 5.3

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для начала работы с программой необходимо открыть exe-файл “MP3 Player”, находящийся в папке “debug” (рис. 6.1).

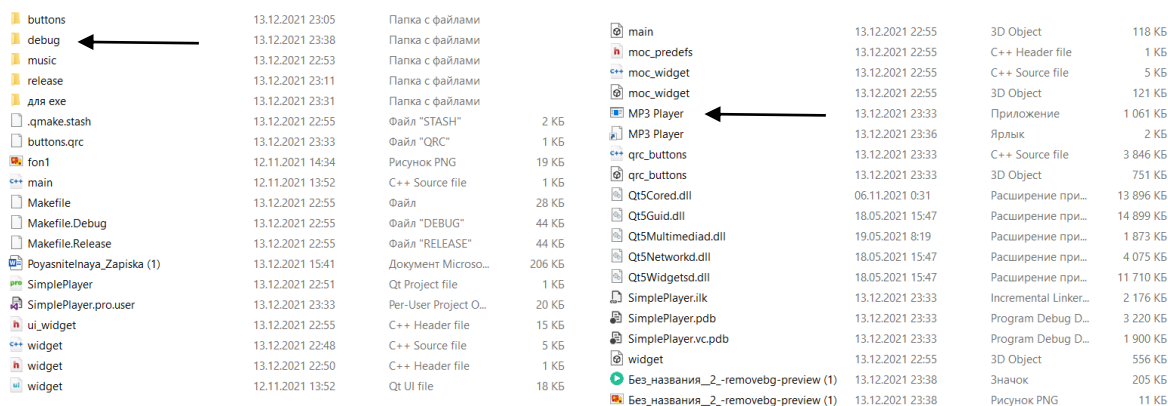


Рисунок 6.1

Запустив этот файл, пользователь увидит главное окно программы, которое состоит из блока плейлиста и блока взаимодействия пользователя с приложением (рис. 6.2).

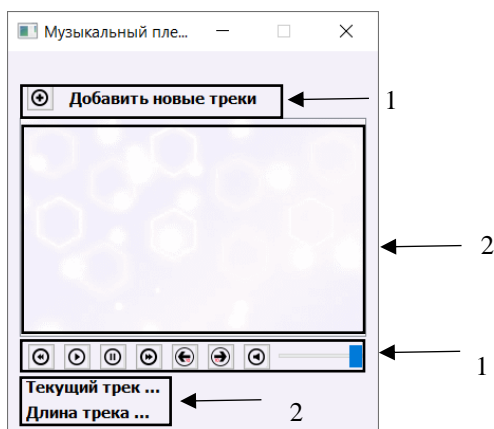



Рисунок 6.2

(1 - блок взаимодействия пользователя с приложением, 2 - блок плейлиста)

Рассмотрим более подробно блок взаимодействия пользователя с приложением. Блок состоит из восьми кнопок и одного слайдера.

Кнопка  служит для добавления одной или нескольких композиций в плейлист. Нажав на кнопку пользователь будут перенаправлен в Проводник

для выбора аудиофайлов. Для добавления аудиофайла в плейлист необходимо дважды щелкнуть левой кнопкой мыши по выбранному файлу. После чего он будет отображаться в верхней части блока плейлиста (рис. 6.3)

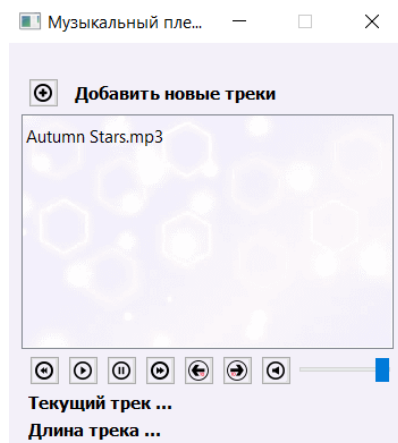


Рисунок 6.3

Для добавления нескольких аудиофайлов в плейлист пользователь должен зажать клавишу Ctrl и одинарными нажатиями левой кнопки или путём зажатия левой кнопки мыши выбрать все нужные файлы, после чего нажать на кнопку “Открыть” (рис. 6.4)

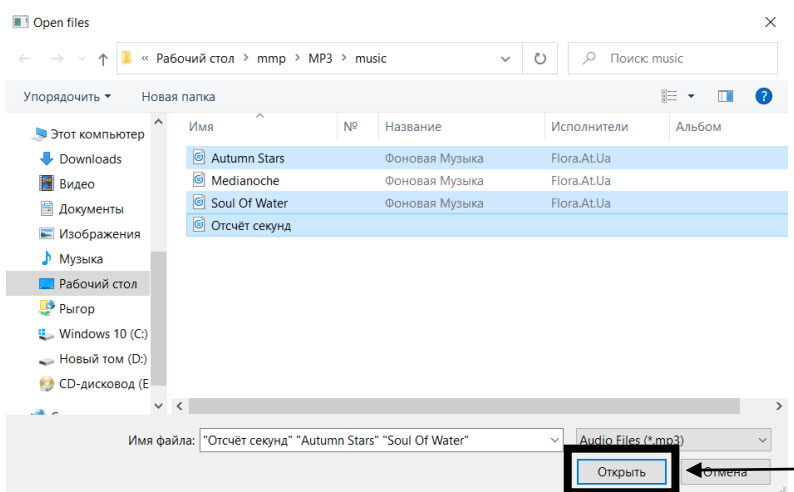



Рисунок 6.4

После того, как файлы будут добавлены в плейлист пользователь может начать прослушивать аудиофайлы путём нажатия левой кнопкой

мышью по названию файла в блоке плейлиста с последующим нажатием кнопки .

Если все вышеописанные действия были выполнены правильно и на компьютере включен звук, то аудиофайл начнёт воспроизводиться, а в нижней части блока плейлиста будет отображено название воспроизводящегося аудиофайла, а также его длина в формате “минуты:секунды” (рис. 6.5).

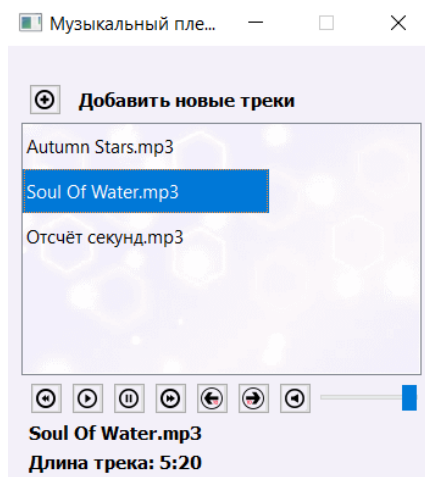











Рисунок 6.5

Если в плейлисте находится два и более аудиофайла, пользователь может переключаться к предыдущему аудиофайлу путём нажатия кнопки , а также к следующему нажатием кнопки .

Кнопка  служит для временной остановки воспроизведения текущего аудиофайла. Продолжить воспроизведение из той же позиции пользователь может нажатием кнопки .

Кнопки  и  служат для перемотки аудиофайла на 10 секунд назад и вперёд соответственно. В случае если кнопка для перемотки назад будет нажата в начале воспроизведения аудиофайла (время воспроизведения будет меньше 10 секунд), аудиофайл начнёт своё воспроизведение с самого начала. В случае, если кнопка для перемотки вперёд будет нажата в конце воспроизведения аудиофайла (оставшееся время воспроизведения меньше 10 секунд), воспроизведение текущего аудиофайла прекратится и сразу же начнёт воспроизводиться следующий аудиофайл в плейлисте.

Кнопка  нужна для выключения звука в приложении. При нажатии на эту кнопку звук в приложении выключится, кнопка изменит своё оформление и станет выглядеть вот так: . Повторным нажатием на кнопку звук включится и кнопка вернётся к стандартному оформлению.

Помимо использования кнопки, пользователь может изменять громкость с помощью слайдера . Чем правее находится синий прямоугольник на слайдере, тем выше громкость воспроизведения.

ЗАКЛЮЧЕНИЕ

В данной работе был проведен анализ поставленной проблемы с последовательным её решением.

Целью разрабатываемого приложения являлась разработка MP3 плеера со стандартным функционалом. В результате проведения работы она была выполнена.

Цель была достигнута путем успешного выполнения основных задач курсовой работы.

Выявленные в ходе тестирования ошибки были успешно устранены. С помощью разработанного MP3 Player можно прослушивать MP3 файлы, менять громкость, перематывать композиции, останавливать воспроизведение. Данный проект может быть усовершенствован в следующих направлениях:

- Реализация эквалайзера - программы, которая позволяет регулировать звук: делать тише или громче определенную звуковую зону, например, какой-либо инструмент в композиции.
- Реализация алгоритма удаления композиции из плейлиста.
- Добавление фоновых картинок для каждой из воспроизводимых композиций.

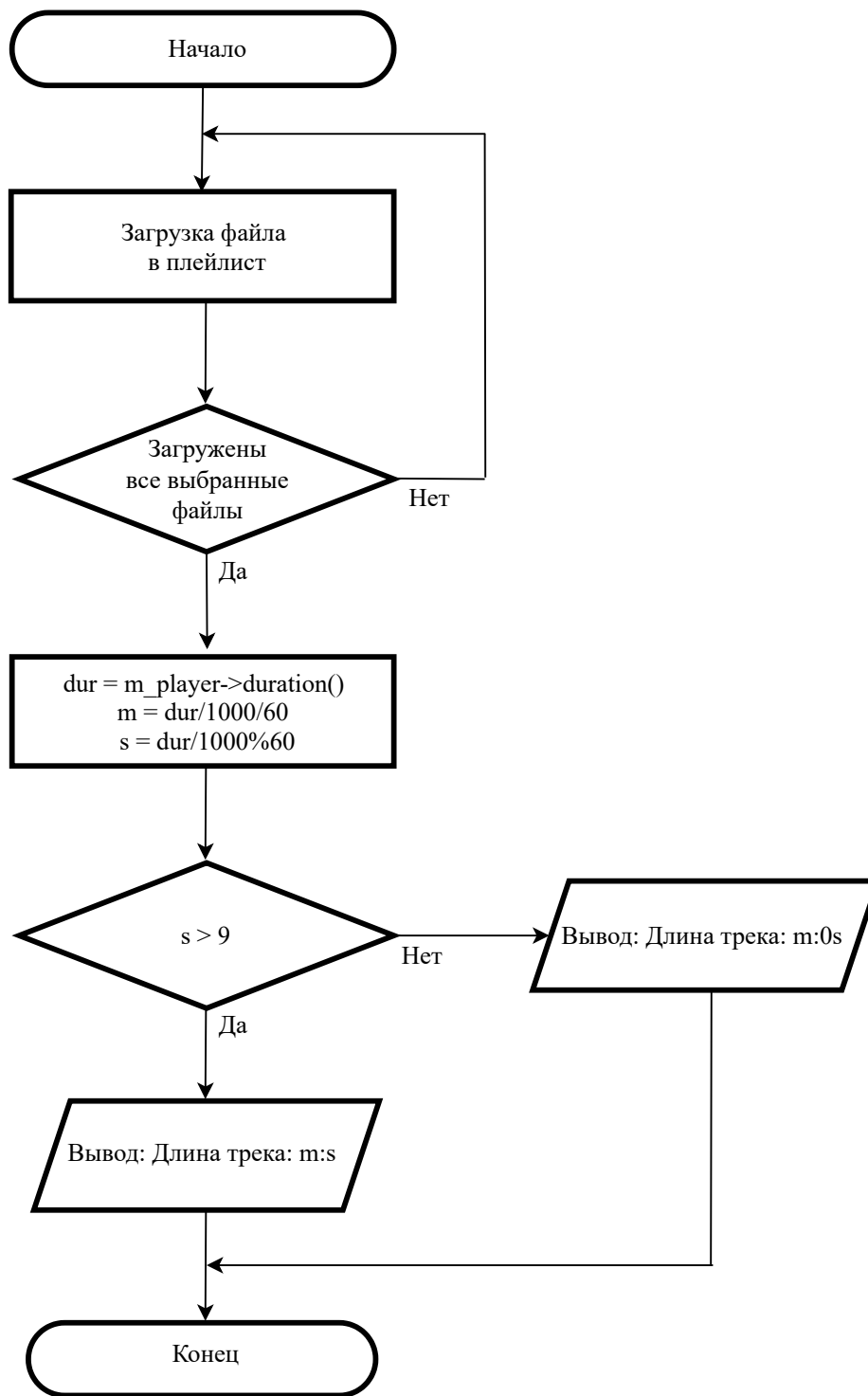
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- 1 Дейтел, Х. М. Как программировать на С++ / Х. М. Дейтел, П. Д. Дейтел; пер. с англ. – М. : Бином, 2007.
- 2 Страуструп, Б. Язык программирования С++ / Б. Страуструп; специальное издание; пер. с англ. – СПб. : BHV, 2008.
- 3 Скляр, В. А. Язык С++ и объектно-ориентированное программирование: справ. пособие / В. А. Скляр. – Минск : Выш. шк., 1997.

ПРИЛОЖЕНИЕ А

(обязательное)

Схема функции `void Ui::Widget::on_btn_add_clicked()`



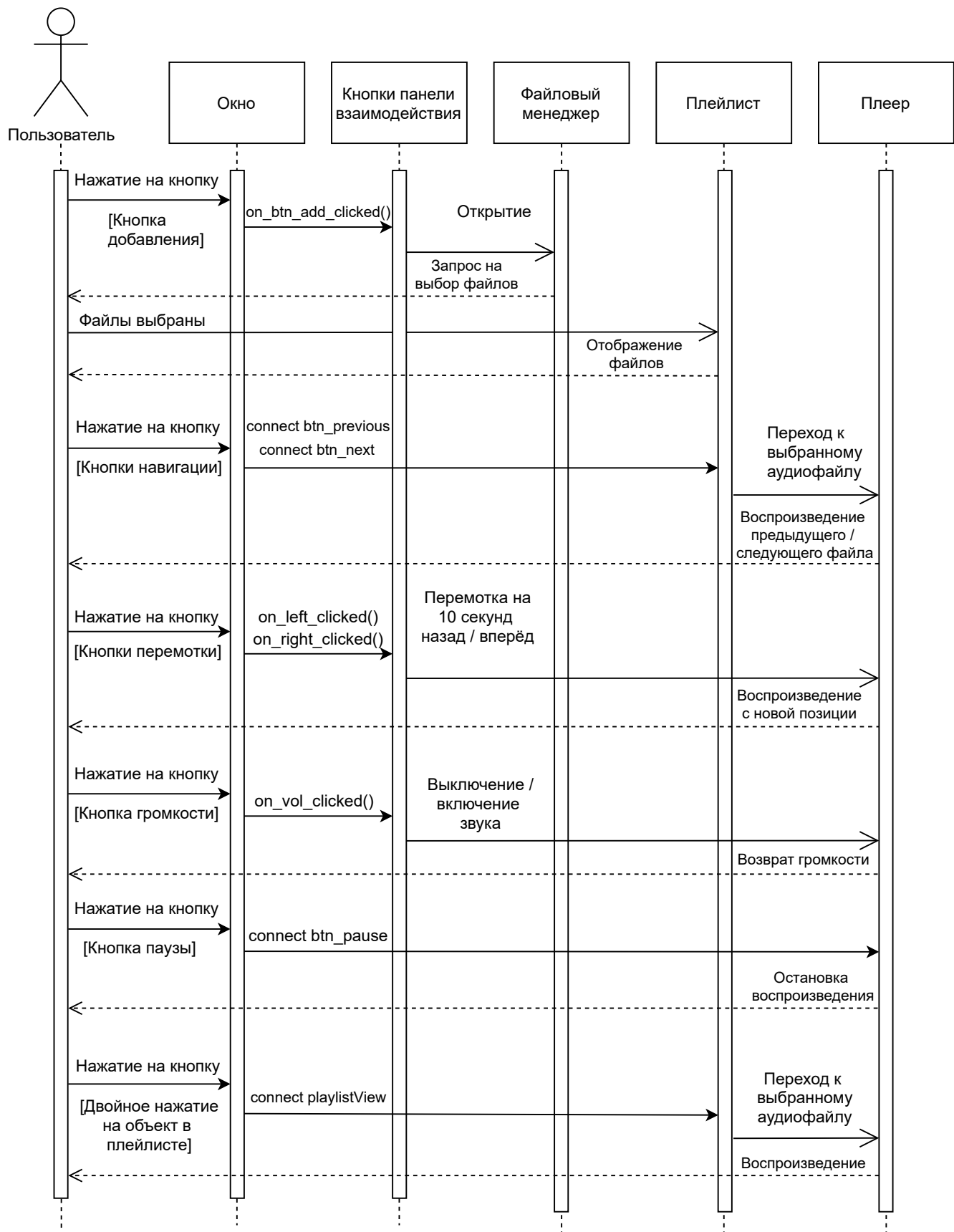
					ГУИР.400201.313 Д1									
					Блок-схема алгоритма					Лит.		Масса	Масштаб	
Изм	Лист	№ документа	Подпись	Дата										
Разраб.	Липский										У			
Пров.	Байдун													
										Лист 1		Листов 1		
										ЭВМ, гр. 050503				

ПРИЛОЖЕНИЕ Б
(обязательное)

Диаграмма классов.

ПРИЛОЖЕНИЕ В
(обязательное)

Диаграмма последовательности.



					ГУИР.400201.313 ДЗ						
					Диаграмма последовательностей приложения						
Изм	Лист	№ документа	Подпись	Дата					Лит.	Масса	Масштаб
Разраб.	Липский								У		
Пров.	Байдун								Лист 1	Листов 1	
					ЭВМ, гр. 050503						