**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 2](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566956)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566957)

[1.1 Обзор области разработки 4](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566958)

[1.2 Обзор существующих продуктов 4](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566959)

[1.2.1 Book Diary 4](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566960)

[1.2.2 StoryGraph:Reading Tracker 5](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566961)

[1.2.3 Goodreads 5](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566962)

[2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 7](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566964)

[2.1 Схема работы системы 7](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566965)

[2.2 Архитектура классов программы 8](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566966)

[3 ПРАКТИЧЕСКАЯ ЧАСТЬ 1](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566967)0

[3.1 Разработка программы 1](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566968)0

[3.2 Тестирование 1](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566969)3

[ЗАКЛЮЧЕНИЕ 1](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566970)5

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 1](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566971)6

[Приложение А. Исходный код программы 1](file:///C:\Users\Dariy\Downloads\Отчёт%20по%20курсовой%20работе.docx#_Toc132566972)7

**ВВЕДЕНИЕ**

Темой работы является разработка приложения "Читательский дневник" на языке C++. Основной идеей работы по данной теме является изучение возможностей языка C++ для решения задачи, имеющей практическую составляющую.

Актуальность работы заключается в том, что разработка приложения "Читательский дневник" на языке С++ не только позволит расширить и улучшить имеющиеся навыки программирования, но также представляет собой отличную возможность создать уникальный продукт, который упростит чтение книг и повысит его эффективность. Такое приложение может стать востребованным среди людей, которые хотят контролировать своё чтение и улучшить свои навыки. Более того, данный проект может оказать полезное влияние на развитие многих читателей, поскольку позволит им более осознанно и целенаправленно подходить к чтению и делать выводы о своих личных предпочтениях и интересах на основе оставленных заметок в приложении.

Цель проекта "Читательский дневник" на языке C++ с использованием фреймворка QT заключается в создании приложения, которое позволит пользователям вести учёт прочитанных книг. Пользователи смогут добавлять в приложение информацию о книгах: название, автора, оценка. Кроме того, приложение должно обеспечивать сохранение данных о книгах и заметках между сеансами использования, чтобы пользователи могли возвращаться к своим записям в любое время

Задачи работы:

1. Провести обзор источников, связанных с темой работы.

2. Изучить несколько существующих решений, которые соответствуют поставленной задаче, и провести их анализ.

3. Составить алгоритм работы программного продукта.

4. Составить иерархию объектов системы.

5. Разработать программу на языке С++, используя полученную информацию.

6. Протестировать разработанный программный продукт, чтобы убедиться в его правильном функционировании и соответствии требованиям.

7. Составить отчет по проведенной работе.

8. Защитить проект путем представления разработанного программного продукта и отчета об его создании.

Чтобы объективно оценить выполненный проект и провести тестирование программы, сформирована таблица критериев достижения успеха (Таблица 1), содержащая десять основных пунктов и представляющая собой макет технического задания в упрощённом виде.

*Таблица 1 — Критерии достижения успеха*

|  |  |  |
| --- | --- | --- |
| № | Критерий | Значение |
| 1 | Функциональность | Кнопки «Добавить  книгу», «Список книг»,  «Ввод данных» |
| 2 | Интерфейс | Есть |
| 3 | Масштабируемость, адаптивность | Есть |
| 4 | Просмотр файла с книгами | Есть |
| 5 | Кроссплатформенность | Windows, Mac, Linux |
| 6 | Загрузка данных | Файл формата .txt |
| 7 | Добавление новых записей в дневник | Есть |
| 8 | Проверки на возможные ошибки | Есть |
| 9 | Оценка книг | Есть |
| 10 | Добавление более трёх книг | Есть |

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ
2. Обзор области разработки

Читательский дневник — это приложение, которое позволяет пользователям записывать информацию о прочитанных книгах. Для создания читательского дневника на языке С++ можно использовать фреймворк Qt. Это позволит быстро и легко создавать кроссплатформенные десктопные приложения с использованием C++. Qt имеет множество библиотек для работы с графическим интерфейсом, базами данных, сетевыми протоколами и многим другим, что значительно упрощает создание приложения. Также, приложение может использовать функционал Qt для создания пользовательского интерфейса. Qt имеет набор виджетов, таких как кнопки, текстовые поля, таблицы и т.д., которые можно использовать для создания интерфейса приложения. В целом, создание Читательского дневника на языке С++ с использованием Qt предоставляет мощный инструментарий для быстрой и качественной разработки десктопных приложений. Кроссплатформенность Qt позволяет создавать приложения, которые будут работать на различных операционных системах, что упрощает процесс распространения и доставки приложения до конечного пользователя.

1. Обзор существующих продуктов

Существует множество приложений, которые помогают сделать чтение более увлекательным и интересным. Рассмотрим несколько таких.

1. Book Diary

Первым идет приложения Book Diary — это приложение, которое позволяет

пользователям вести дневник прочитанных книг. В нем возможно добавление

книги в дневник с информацией о названии, авторе, издании, жанре, дате

прочтения. Эффективно распределяет ресурсы устройства, в том числе

память.Также оно обладает хорошим подбором интересующих книг и

качественной статистикой. Ну и конечно отличным интуитивным дизайном. Но в Book Diary не такая большая база данных книг, что усложняет процесс их

добавления.

1. StoryGraph: Reading Tracker

StoryGraph: Reading Tracker — это онлайн-сервис для отслеживания прочитанных книг и управления своей личной библиотекой. Этот сервис позволяет пользователям добавлять книги, которые они прочитали или планируют прочитать в будущем, а также устанавливать сроки чтения и отслеживать свой прогресс в чтении, что способствует поддержание чтения. Одним из преимуществ StoryGraph перед другими сервисами для чтения является его фокус на личной статистике чтения и управлении библиотекой, что делает его особенно привлекательным для любителей чтения. Также, благодаря возможности экспорта данных, пользователи могут сохранить свои прочтения и на своем компьютере или в другом месте. Однако на данный момент сообщество этого приложения уступает другим.

1. Goodreads

И последнее - Goodreads. Это социальная сеть для книголюбов, которая позволяет пользователям оценивать и обсуждать книги, составлять свои списки прочитанного, а также находить новые книги на основе своих интересов. В приложении можно создавать профиль, добавлять друзей и следить за их чтением, получать персональные рекомендации на основе своей истории чтения, участвовать в челленджах и группах, где пользователи обсуждают книги по разным тематикам. Также есть функции отслеживания прогресса чтения и составления списков желаемых книг, что стимулирует продолжать чтение книг. Goodreads является одной из самых популярных платформ для общения о книгах, что означает наличие крупного комьюнити. Но в отличии от предыдущих приложений Goodreads не имеет встроенного инструмента для отслеживания прочитанных страниц и времени чтения, а также присутствуют менее точные рекомендации на основе истории чтения.

1. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ
2. **Схема работы системы**

После запуска десктоп приложения пользователь видит перед собой

главное меню, которое состоит из трех кнопок: "Добавить книгу", "Список книг" и "Выход".

При нажатии на кнопку "Добавить книгу" появляется окно ввода, где пользователь может ввести информацию о новой книге, такую как: название, автора и оценку прочитанной книги. После ввода данных пользователь должен

нажать кнопку "Добавить", чтобы сохранить информацию в txt файле.

При нажатии на кнопку "Список книг" открывается окно с полным списком всех добавленных пользователем книг.

При нажатии на кнопку "Выход" появляется окно с предупреждением. Если пользователь подтверждает свое желание выйти из приложения, то происходит завершение работы приложения. Если пользователь отменяет выход, он остается в приложении и может продолжить работу с ним.

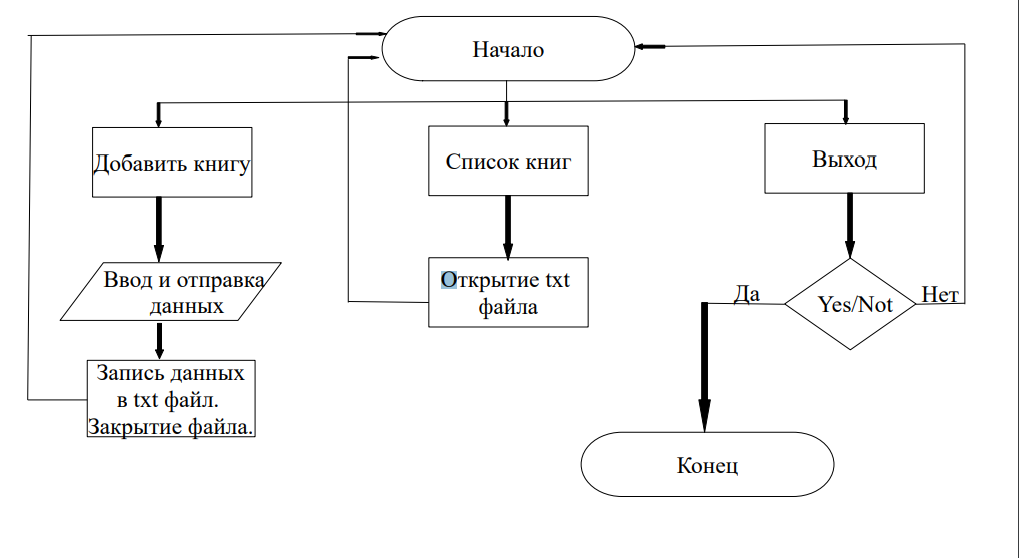


Рисунок 2.1 — Блок схема работы игры

1. **Архитектура классов программы**

В данной программе используется большое количество классов. Некоторые из них представлены в таблице и схеме. Класс Book является подклассом QDialog и содержит следующие элементы:

*Таблица 2.1*

|  |  |
| --- | --- |
| Свойства, классы и методы | Описание |
| Book(QWidget \*parent = nullptr) | Конструктор, который принимает  родительский виджет в качестве аргумента |
| ~Book() | Деструктор, который освобождает любые  выделенные ресурсы, связанные с объектом. |
| on\_checkBox\_clicked() | Приватный слот, который вызывается при  клике на флажок. |
| on\_pushButton\_clicked() | Приватный слот, который вызывается при  клике на кнопку QPushButton. |
| on\_pushButton\_2\_clicked() | Приватный слот, который вызывается при  клике на другую кнопку QPushButton. |
| on\_But\_clicked() | Приватный слот, который вызывается при  клике на еще одну кнопку QPushButton |
| Ui::Book \*ui | Приватный член Ui::Book \*ui, который  является указателем на экземпляр класса  Ui::Book, содержащий пользовательский  интерфейс для этой формы. |

Класс MainWindow является подклассом QMainWindow и содержит

следующие элементы:

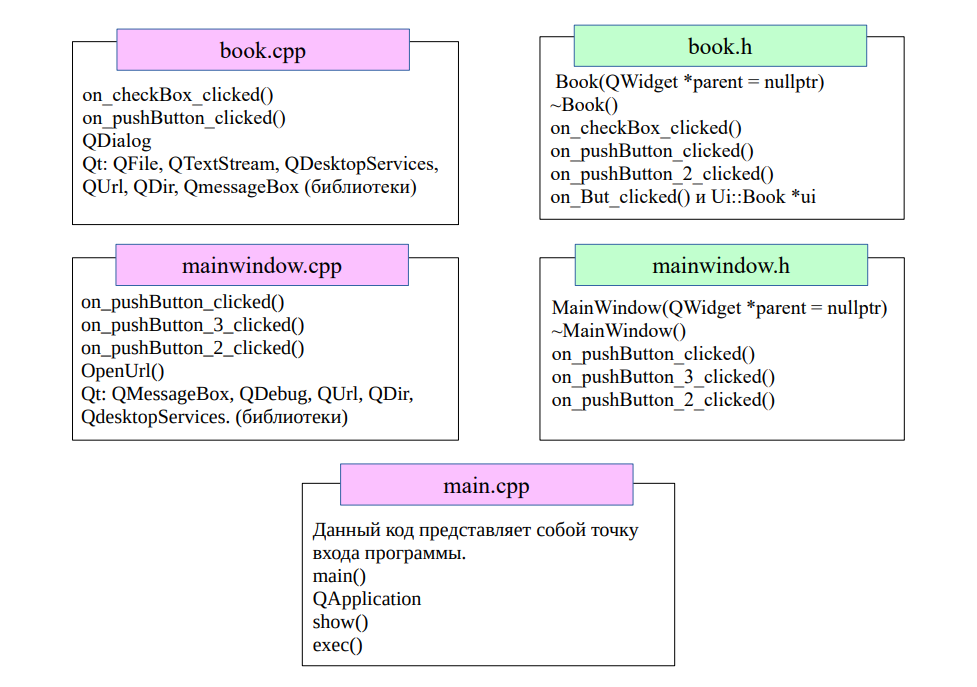
*Таблица 2.2*

|  |  |
| --- | --- |
| Свойства, классы и методы | Описание |
| MainWindow(QWidget \*parent = nullptr) | Конструктор класса, который устанавливает  заголовок окна и вызывает метод setupUi() для  создания пользовательского интерфейса. |
| ~MainWindow() | Деструктор класса, который освобождает  память, занятую объектом ui. |
| on\_pushButton\_clicked() | Обработчик события нажатия на кнопку  "Добавить книгу". Создает объект класса Book,  устанавливает его модальным и вызывает  метод exec(), чтобы открыть его в модальном  режиме. |
| on\_pushButton\_3\_clicked() | Обработчик события нажатия на кнопку  "Выход". Отображает диалоговое окно с  вопросом о выходе из приложения. Если  пользователь ответил "Да", то вызывается  метод quit() объекта QApplication, чтобы  закрыть приложение. |
| on\_pushButton\_2\_clicked() | Обработчик события нажатия на кнопку  "Открыть файл". Создает объект типа QUrl,  который ссылается на файл books.txt в текущей  директории, а затем вызывает статический  метод openUrl() класса QDesktopServices,  чтобы открыть файл в соответствующем  приложении. |

Функция main, которая является точкой входа в программу. Она начинает выполнение программы, создает экземпляр класса QApplication и экземпляр класса MainWindow, после чего отображает главное окно приложения с помощью метода show(). Затем, функция a.exec() запускает цикл обработки событий QApplication и ожидает, пока пользователь не закроет приложение. После этого программа завершается с возвращением кода выхода. Таким образом, архитектура классов для данного кода включает:

*Таблица 2.3*

|  |  |
| --- | --- |
| Классы | Описание |
| QApplication | Используется для управления основными  системными ресурсами приложения. |
| MainWindow | Главное окно приложения, которое  наследуется от QMainWindow или QDialog,  содержит интерфейс пользователя и логику  работы приложения. |



**Рисунок 2.2 — Схема архитектуры классов программы**

1. ПРАКТИЧЕСКАЯ ЧАСТЬ
2. **Разработка программы**

Работа с десктоп приложением начинается с запуска. Перед пользователем сразу появляется Меню.

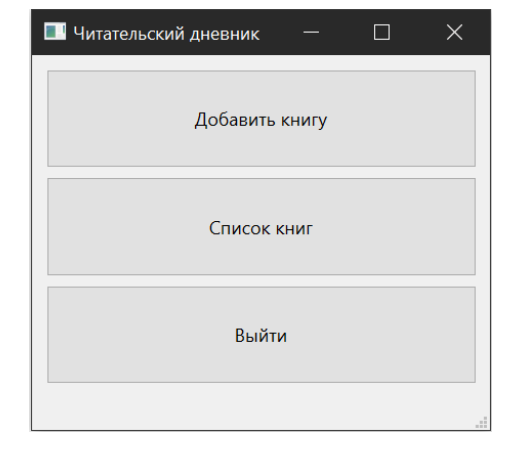
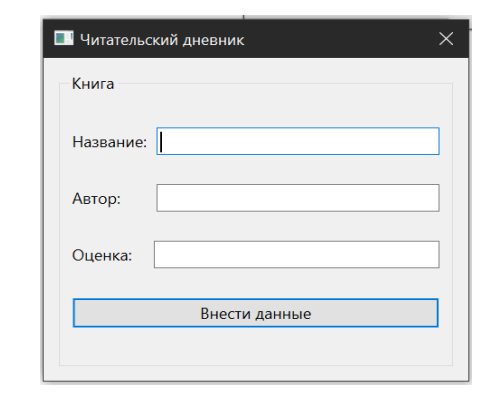


Рисунок 3.1 — Меню программы

В данном меню представлены три кнопки, такие как «Добавить книгу», «Список книг», «Выход». При нажатии на кнопку «Добавить книгу» появляется новое окошко с полями для ввода данных.



**Рисунок 3.2 — Функция «Добавить книгу»**

Пользователь вводит «Название», «Автор», «Оценка» и нажимает кнопку «Внести данные», чтобы данные появились в txt файле. При этом если пользователь оставит какое-то поле пустым, то вылезет окно предупреждающие об ошибке.

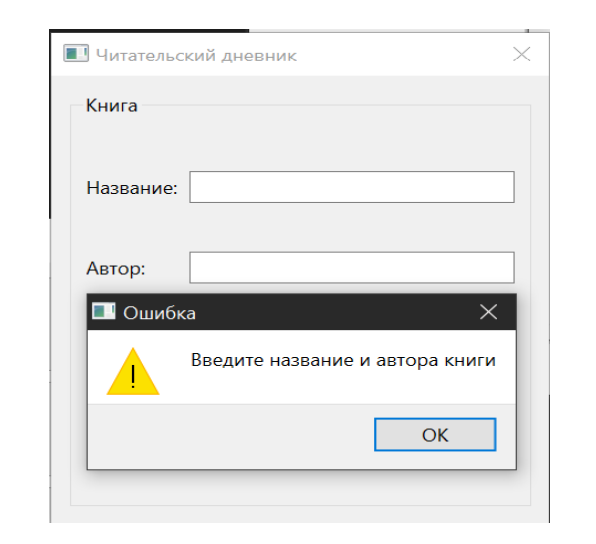
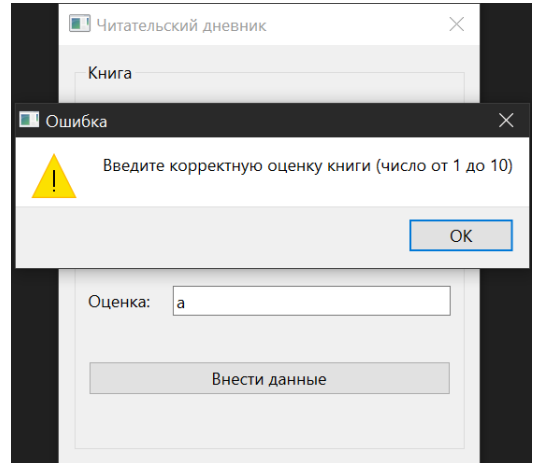


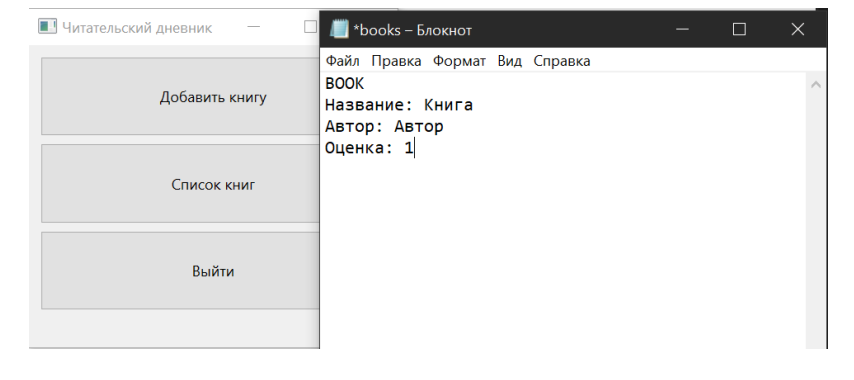
Рисунок 3.3 — Проверка заполнения полей

Если пользователь решит ввести в поле «Оценка» что-то вместо цифр (от 0 до 10), то вылезет окно предупреждающие об ошибке.



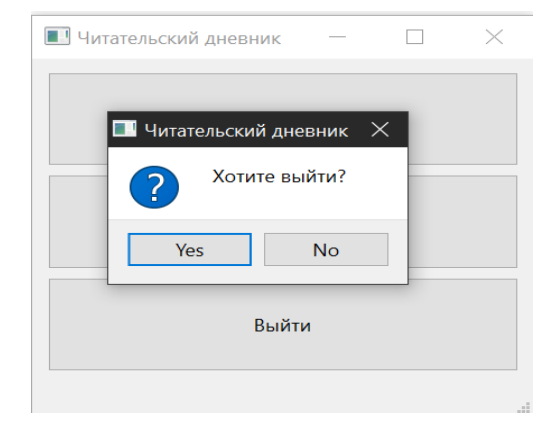
**Рисунок 3.4 — Проверка корректности вводных данных**

После того как пользователь вводит все данные верно, то они сразу вносятся в txt файл и сохраняются там. Чтобы посмотреть список пользователь должен нажать кнопку «Список книг», после чего открывается txt файл с записями.



**Рисунок 3.5 — Функция «Список книг»**

В данном файле будут храниться все записи о книгах. Чтобы выйти из приложения достаточно нажать кнопку «Выход».



**Рисунок 3.6 — Функция «Выйти»**

После нажатия вылезает окошко, в котором мы выбираем «Yes» или «No». При нажатии кнопки «Yes» программа закрывается.

**3.2 Тестирование**

Для тестирования программы сформирован журнал испытаний программы (Таблица 3.1). Итоговое тестирование показало, что функционал программы реализован и работает корректно.

*Таблица 3.1 — Журнал испытаний программы*

|  |  |  |  |
| --- | --- | --- | --- |
| № испытания | Испытание | Эталонный результат | Фактический результат |
| 1 | Запуск программы | Открывается окно с меню программы | Открывается окно с меню программы |
| 2 | Добавление книги | Появляются строки для ввода автора, названия и оценки | Появляются строки для ввода автора, названия и оценки |
| 3 | Сохранение данных без заполнения одного и более полей | При попытке сохранения данных без заполнения одного и более полей, программа выдает ошибку | При попытке сохранения данных без заполнения одного и более полей, программа выдает ошибку |
| 4 | Ввод нечисловых значений в поле «оценка» | Выводится сообщение о некорректном вводе | Выводится сообщение о некорректном вводе |
| 5 | Сохранение данных | Данные вносятся в txt файл и сохраняются | Данные вносятся в txt файл и сохраняются |
| 6 | Просмотр «Списка книг» | Открывается txt файл с записями | Открывается txt файл с записями |
| 7 | «Выход» из программы | Появляется окно, в котором мы выбираем «Yes» или «No». При нажатии кнопки «Yes» программа закрывается | Появляется окно, в котором мы выбираем «Yes» или «No». При нажатии кнопки «Yes» программа закрывается |

ЗАКЛЮЧЕНИЕ

Результатом работы является выполнение следующих задач:

1. Был проведен обзор источников, важных для работы.

2. Изучены несколько существующих решений, которые соответствуют

поставленной задаче, и провести их анализ.

3. Составлен алгоритм работы программного продукта.

4. Составлена иерархию объектов системы.

5. Разработана программа на языке С++, с помощью полученной информации.

6. Протестирован разработанный программный продукт, проверены функционал и соответствие с требованиями.

7. Составлен отчет по проведенной работе.

Приложение "Читательский дневник" на языке C++ с использованием

фреймворка QT прошло проверку, функционал соответствует поставленным задачам. Цель работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальная документация по QT - https://doc.qt.io/qt-5/index.html (дата

обращения 23.05.2023)

2. Документация по языку С++ - https://en.cppreference.com/w/ (дата

обращения: 26.04.2023)

3. Матренин, П. В. Введение в кроссплатформенное программирование на С++ в Qt / П.В. Матренин. - М.: Нобель Пресс, 2013. - 813 c. (дата обращение: 25.05.2023)

4. Винокуров, Игорь Использование библиотеки классов Trolltech Qt для разработки графического интерфейса пользователя / Игорь Винокуров. - Москва: Мир, 2014. - 191 c. (дата обращения: 25.05.2023)

5. Кёнег, Э Эффективное программирование на C++. Практическое программирование на примерах. Серия "C++ In-Depth" / Э Кёнег, Б. Му. - М.: Диалектика, 2019. - 368 c. (дата обращения: 15.05.2023)

**Приложение А**

Исходный код программы

*Листинг А.1 — Файл book.cpp*

#include "book.h"

#include "ui\_book.h"

#include <QFile>

#include <QTextStream>

#include <QDesktopServices>

#include <QUrl>

#include <QDir>

#include <QDebug>

#include <QMessageBox>

Book::Book(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Book)

{

ui->setupUi(this);

this->setWindowTitle("Читательский дневник");

}

Book::~Book()

{

delete ui;

}

void Book::on\_checkBox\_clicked()

{

}

void Book::on\_pushButton\_clicked()

{

const QString bookTitle = ui->namebook->text();

const QString authorName = ui->authorbook->text();

const QString bookGradeString = ui->gradebook->text();

// Проверяем корректность ввода данных

bool isBookGradeValid = false;

const int bookGrade = bookGradeString.toInt(&isBookGradeValid);

if (!isBookGradeValid || bookGrade < 1 || bookGrade > 10) {

QMessageBox::warning(this, "Ошибка", "Введите корректную оценку книги (число от 1 до 10)");

return;

}

// Проверяем, что пользователь ввел Название и Автор книги

if (bookTitle.isEmpty() || authorName.isEmpty()) {

QMessageBox::warning(this, "Ошибка", "Введите название и автора книги");

return;

}

QFile file("books.txt");

if (!file.open(QIODevice::Append | QIODevice::Text)) {

QMessageBox::critical(this, "Ошибка", QString("Не удалось открыть файл %1 для записи").arg(file.fileName()));

return;

}

QTextStream out(&file);

out << "BOOK\n"

<< "Название: " << bookTitle << "\n"

<< "Автор: " << authorName << "\n"

<< "Оценка: " << bookGrade << "\n";

if (!file.flush()) {

QMessageBox::critical(this, "Ошибка", QString("Не удалось записать данные в файл %1").arg(file.fileName()));

} else {

hide();

}

}

*Листинг А.2 — Файл book.h*

#ifndef BOOK\_H

#define BOOK\_H

#include <QDialog>

namespace Ui {

class Book;

}

class Book : public QDialog

{

Q\_OBJECT

public:

explicit Book(QWidget \*parent = nullptr);

~Book();

private slots:

void on\_checkBox\_clicked();

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_But\_clicked();

private:

Ui::Book \*ui;

};

#endif // BOOK\_H

*Листинг А.3 — Файл book.ui*

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>Book</class>

<widget class="QDialog" name="Book">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>300</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Dialog</string>

</property>

<layout class="QVBoxLayout" name="verticalLayout\_2">

<item>

<widget class="QGroupBox" name="groupBox">

<property name="title">

<string>Книга</string>

</property>

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<layout class="QHBoxLayout" name="horizontalLayout">

<item>

<widget class="QLabel" name="line1">

<property name="text">

<string>Название:</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="namebook"/>

</item>

</layout>

</item>

<item>

<layout class="QHBoxLayout" name="horizontalLayout\_2">

<item>

<widget class="QLabel" name="line2">

<property name="text">

<string>Автор: </string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="authorbook"/>

</item>

</layout>

</item>

<item>

<layout class="QHBoxLayout" name="horizontalLayout\_3">

<item>

<widget class="QLabel" name="line3">

<property name="text">

<string>Оценка: </string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="gradebook"/>

</item>

</layout>

</item>

<item>

<widget class="QPushButton" name="pushButton">

<property name="text">

<string>Внести данные</string>

</property>

</widget>

</item>

</layout>

</widget>

</item>

</layout>

</widget>

<resources/>

<connections/>

</ui>

*Листинг А.4 — Файл main.cpp*

#include "mainwindow.h"

#include <QApplication>

#include <QLabel>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

*Листинг А.5 — Файл mainwindow.cpp*

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "book.h"

#include <QMessageBox>

#include <QDebug>

#include <QUrl>

#include <QDir>

#include <QDesktopServices>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

this->setWindowTitle("Читательский дневник");

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked() // Кнопка ДОБАВИТЬ КНИГУ

{

Book book;

// hide();

book.setModal(true);

book.exec();

}

void MainWindow::on\_pushButton\_3\_clicked() // Кнопка ВЫХОД

{

QMessageBox::StandardButton Exit = QMessageBox::question(this, "Читательский дневник", "Хотите выйти?",QMessageBox::Yes | QMessageBox::No);

if(Exit == QMessageBox::Yes){

QApplication::quit();

} else {

qDebug() << "";

}

}

void MainWindow::on\_pushButton\_2\_clicked()

{

QUrl fileUrl = QUrl::fromLocalFile(QDir::currentPath() + "/books.txt");

QDesktopServices::openUrl(fileUrl);

}

*Листинг А.6 — Файл mainwindow.h*

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_2\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

*Листинг А.7 — Файл mainwindow.ui*

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>298</width>

<height>244</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<widget class="QPushButton" name="pushButton">

<property name="sizePolicy">

<sizepolicy hsizetype="Minimum" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="styleSheet">

<string notr="true"/>

</property>

<property name="text">

<string>Добавить книгу</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="pushButton\_2">

<property name="sizePolicy">

<sizepolicy hsizetype="Minimum" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="text">

<string>Список книг</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="pushButton\_3">

<property name="sizePolicy">

<sizepolicy hsizetype="Minimum" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="text">

<string>Выйти</string>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QStatusBar" name="statusbar"/>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>298</width>

<height>17</height>

</rect>

</property>

</widget>

</widget>

<resources/>

<connections/>

</ui>