*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(национальный исследовательский университет)***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_Компьютерные Системы и сети (ИУ6)\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по домашней работе № 3**

**Название домашней работы: Программирование на С++**

**Дисциплина: Объектно-ориентированное программирование**

Студент гр. ИУ6-22Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Плютто**



(Подпись, дата) (И.О. Фамилия)



Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. А. Веселовская**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

***Задание 1. Наследование и композиция. Оформление программной документации средствами doxygen***

Разработать и реализовать диаграмму классов для описанных объектов предметной области, используя механизмы наследования и композиции. Протестировать все методы. Поля классов должны быть частными (private) или защищенными (protected). Методы не должны содержать операций ввода/вывода, за исключением процедур, единственной задачей которых является вывод информации об объекте на экран. Подготовить документацию средствами doxygen.

Объект – заказное письмо. Поля: адрес отправителя, адрес получателя, вес. Методы: процедура инициализации полей, процедура вывода информации о письме на экран, функция вычисления стоимости отправки письма по России согласно текущим почтовым тарифам.

Объект – ценное письмо. Поля: адрес отправителя, адрес получателя, вес, объявленная ценность письма в рублях. Методы: процедура инициализации полей, процедура вывода информации о письме на экран, функция вычисления доплаты за объявленную ценность и функция вычисления полной стоимости отправки письма по России согласно текущим почтовым тарифам.

Объект – почтовый ящик. Содержит в себе ценные письма. Методы объекта должны позволять: инициализировать объект, выводить на экран содержимое ящика, определять суммарную стоимость писем в ящике.

#include "iostream"

#include "string"

#include "vector"

using std::string, std::vector, std::cout, std::cin;

class Letter

{

protected:

string mail1;

string mail2;

int weight;

public:

Letter(string m1, string m2, int w)

{

mail1 = m1;

mail2 = m2;

weight = w;

};

void print()

{

cout << "\nОТПРАВЛЕНИЕ\n------------\nИЗ: " << mail1

<< "\nВ: " << mail2 << "\nВЕС: " << weight;

};

int cost()

{

return weight \* 2.5;

};

};

class CostlyLetter : public Letter

{

protected:

int costs;

public:

CostlyLetter(string m1, string m2, int w, int c) : Letter(m1, m2, w)

{

costs = c;

};

void print()

{

cout << "\nОТПРАВЛЕНИЕ\n------------\nИЗ: " << mail1

<< "\nВ: " << mail2 << "\nВЕС: " << weight

<< "\nЦЕННОСТЬ: " << costs;

};

int costAll()

{

return cost() + costs;

};

};

class Mail

{

protected:

vector<CostlyLetter> letters;

public:

Mail(vector<CostlyLetter> l)

{

letters = l;

}

void print()

{

for (auto i : letters)

i.print();

}

int cost()

{

int res{};

for (auto i : letters)

res += i.costAll();

return res;

}

};

int main()

{

Letter ex1("Moscow", "SPB", 100);

CostlyLetter ex2("SPB", "Moscow", 100, 1000);

Mail ex3({ex2, ex2, ex2});

ex1.print();

cout << "\n"

<< ex1.cost() << "\n";

ex2.print();

cout << "\n"

<< ex2.cost() << "\n"

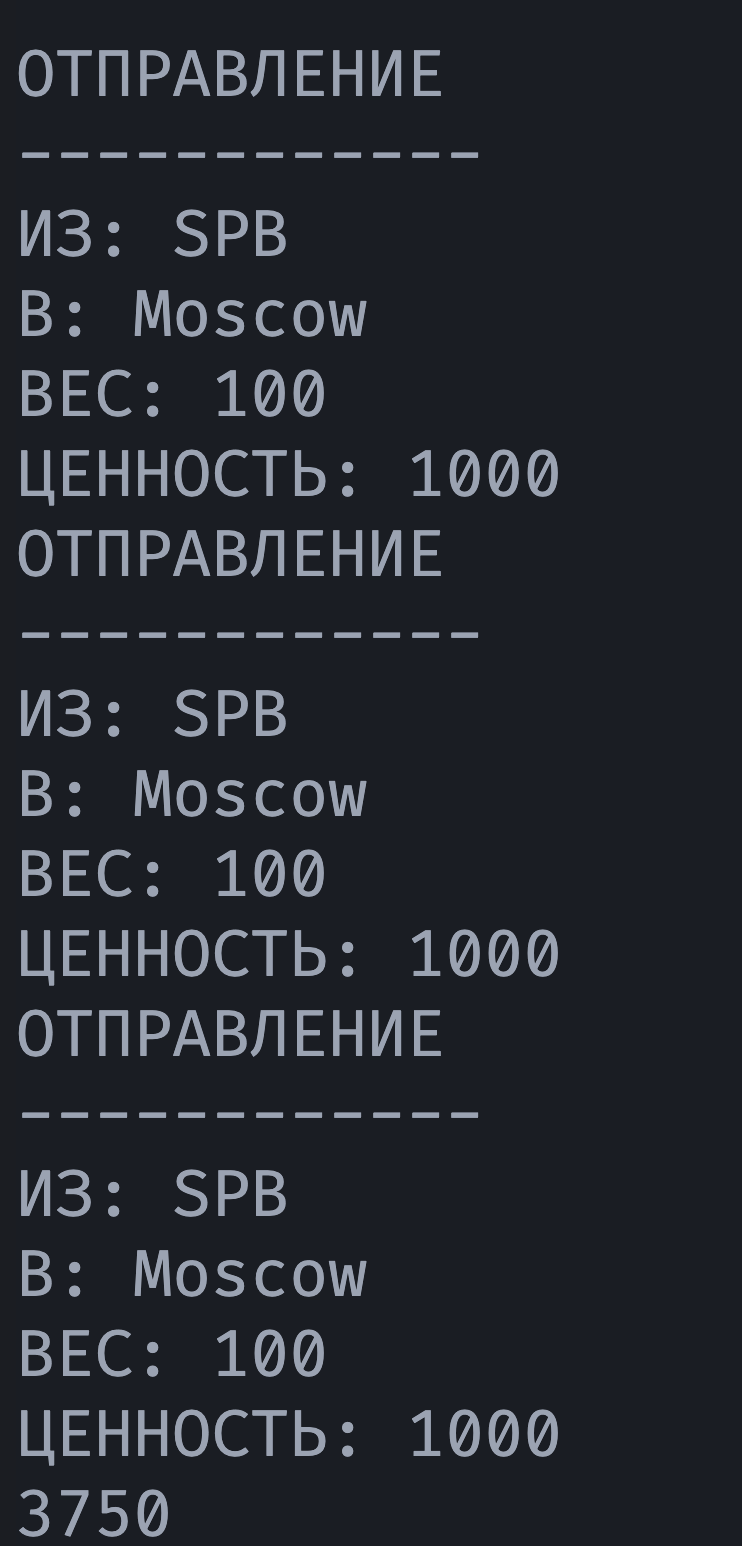
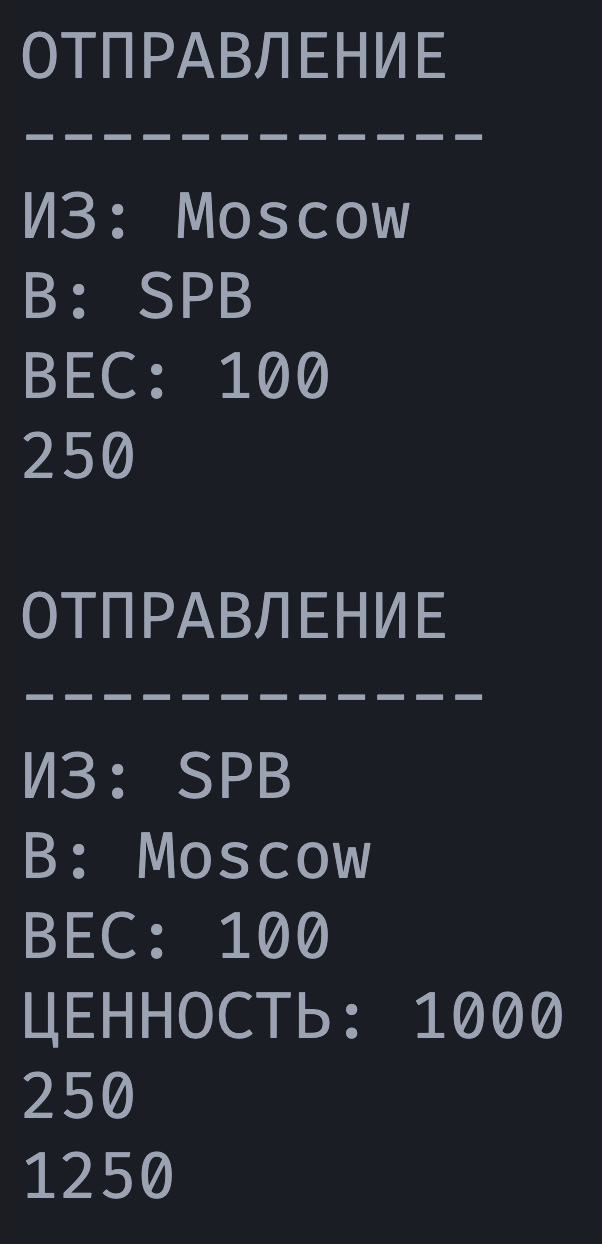
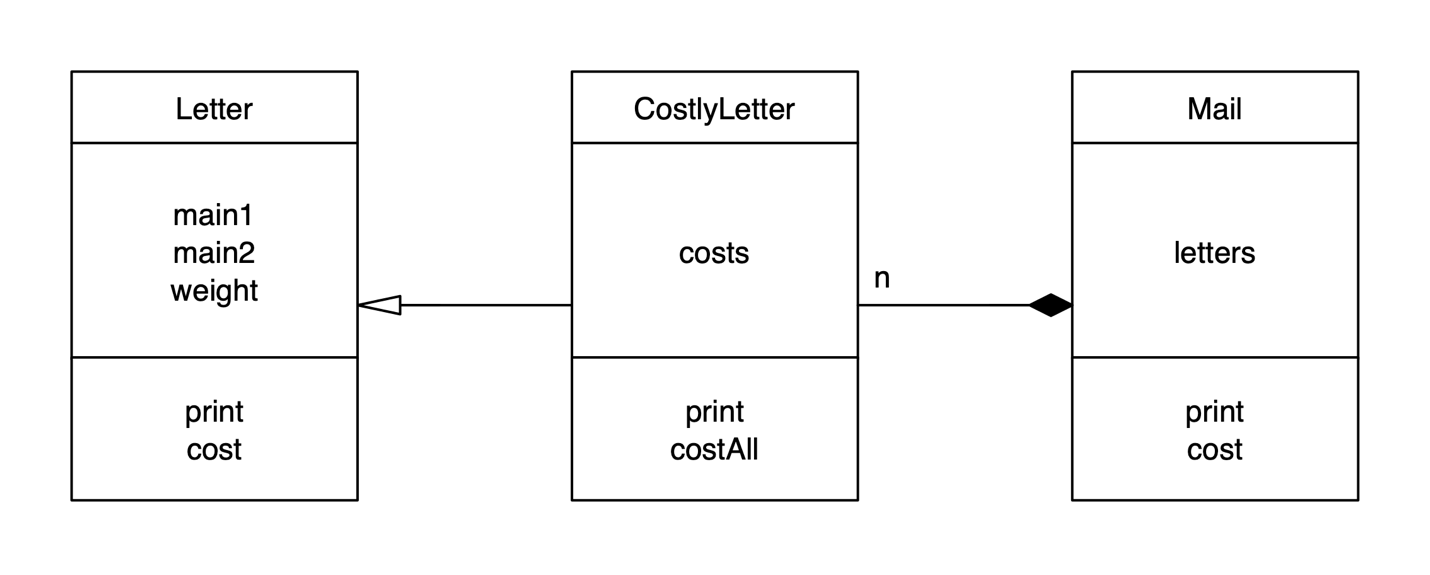
<< ex2.costAll() << "\n";

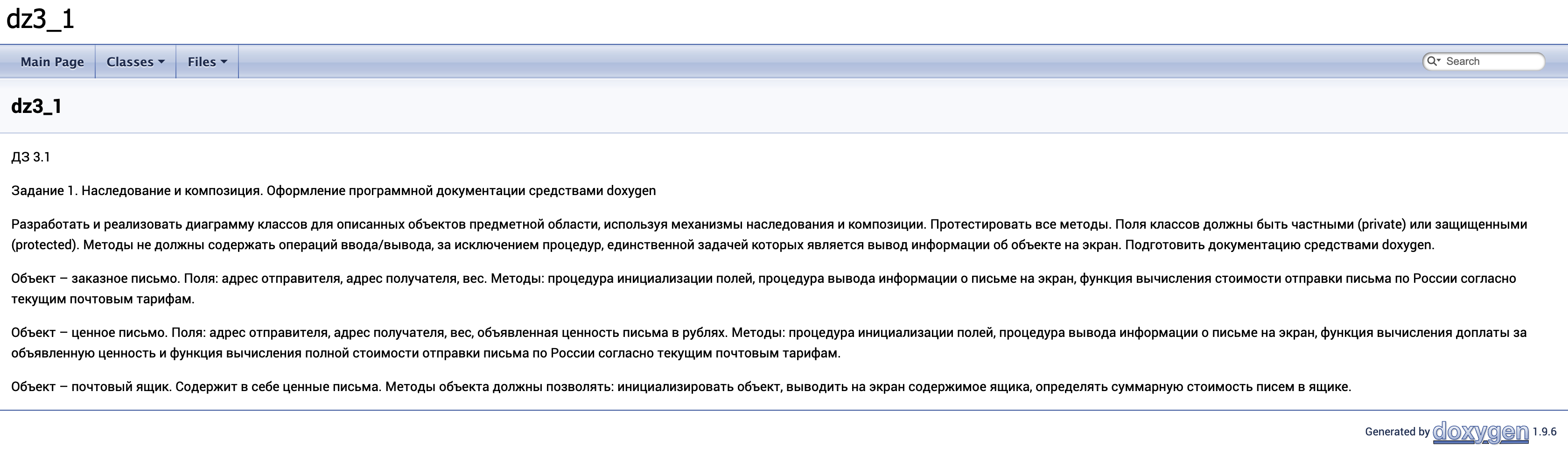
ex3.print();

cout << "\n"

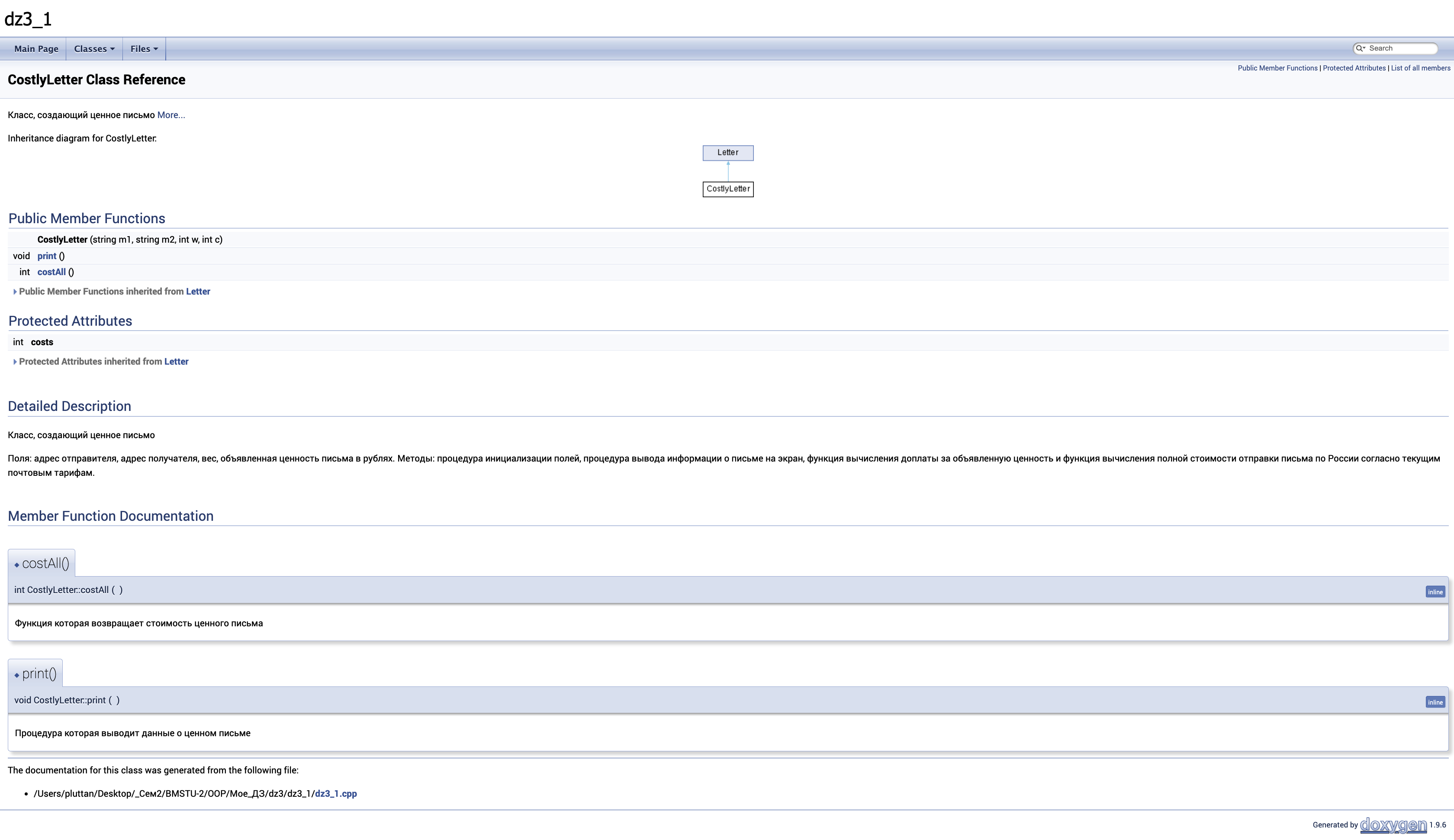
<< ex3.cost() << "\n";

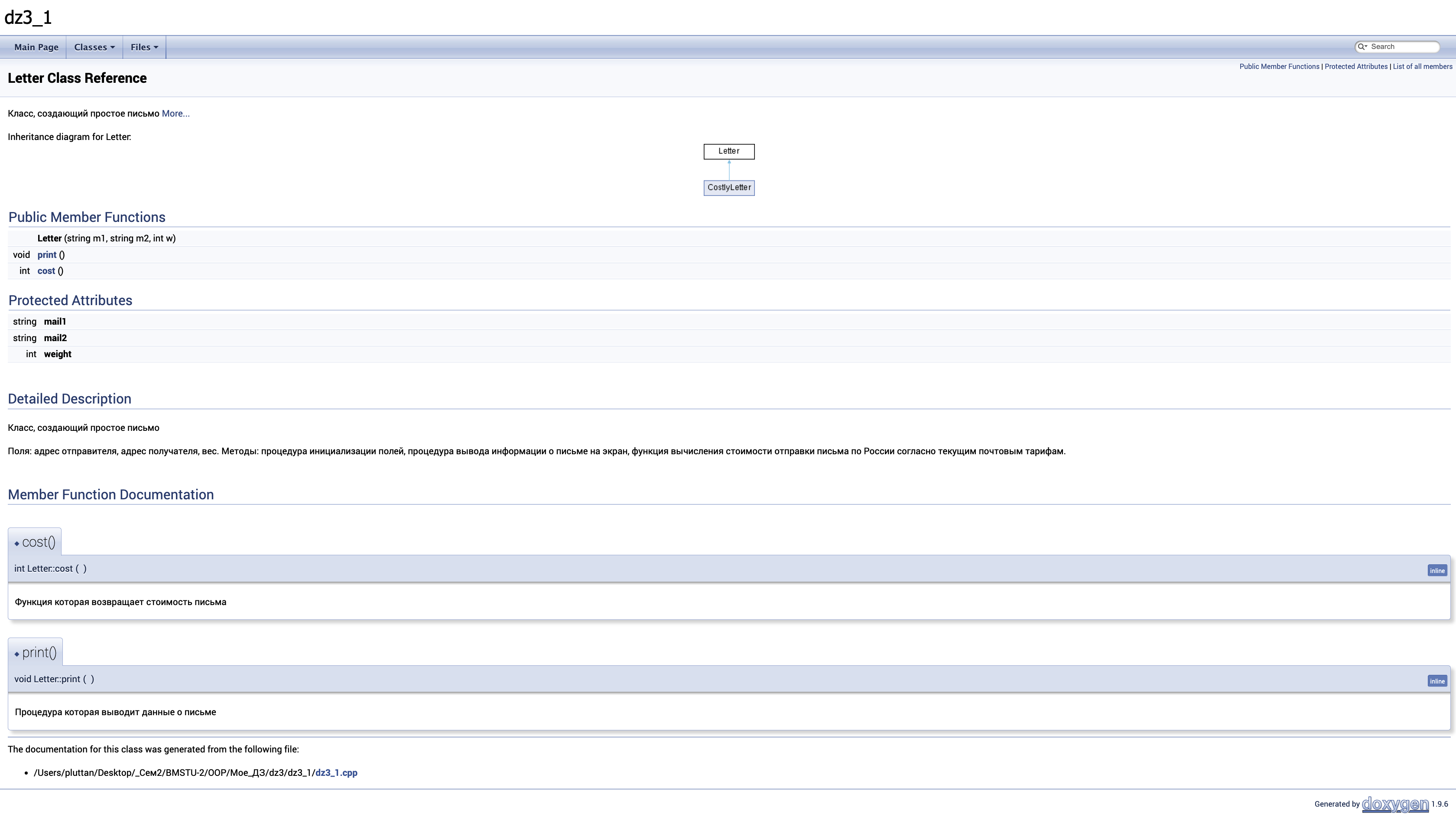
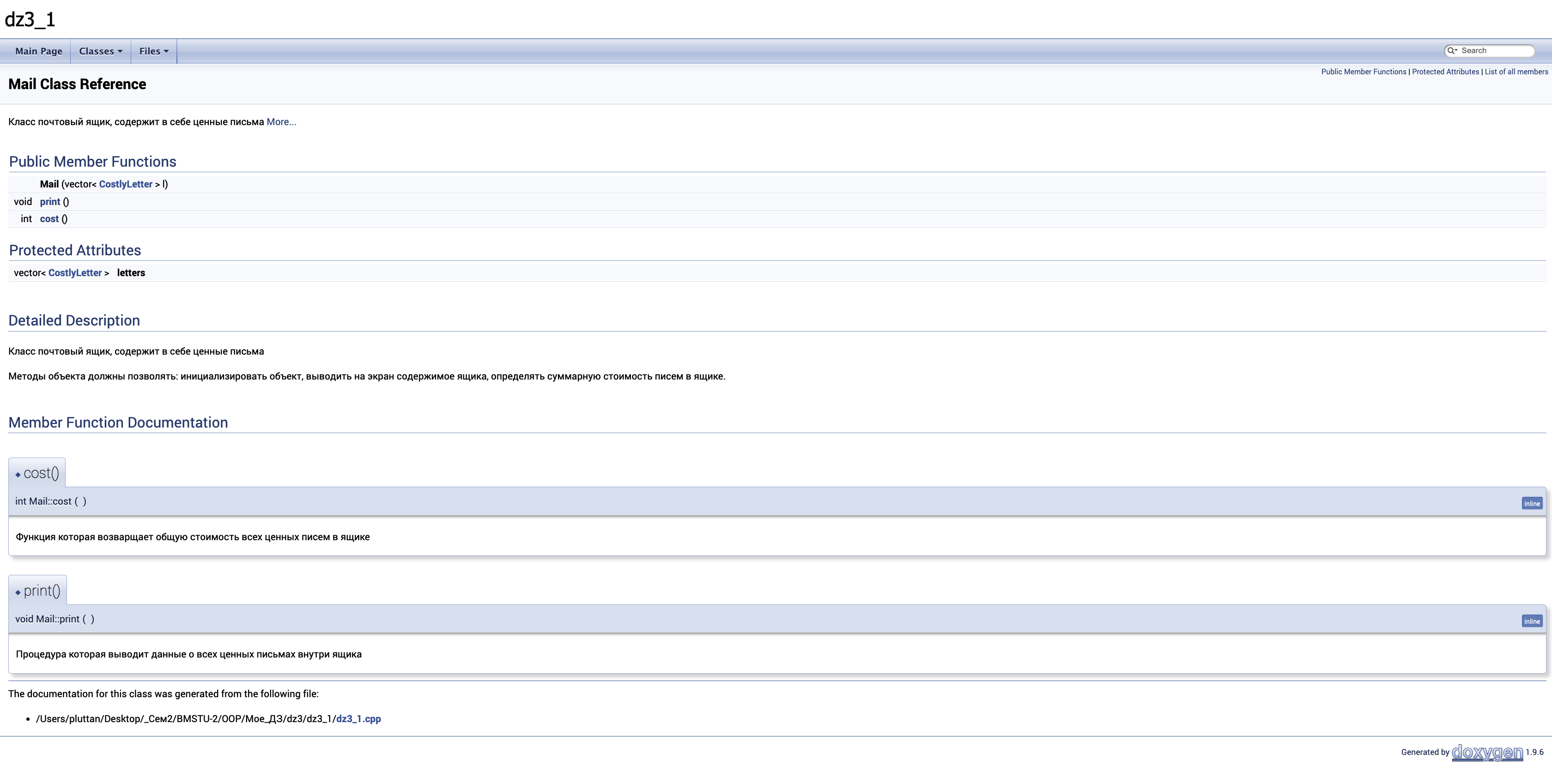
}

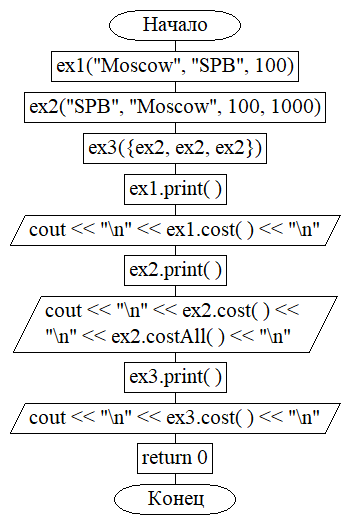
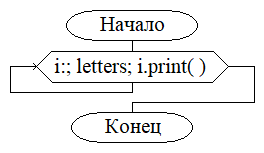
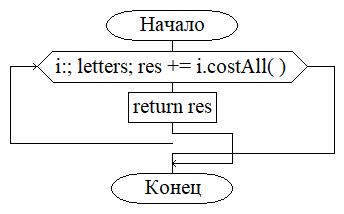
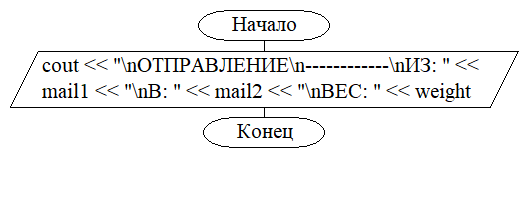
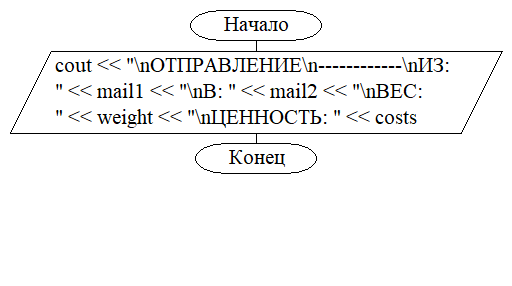
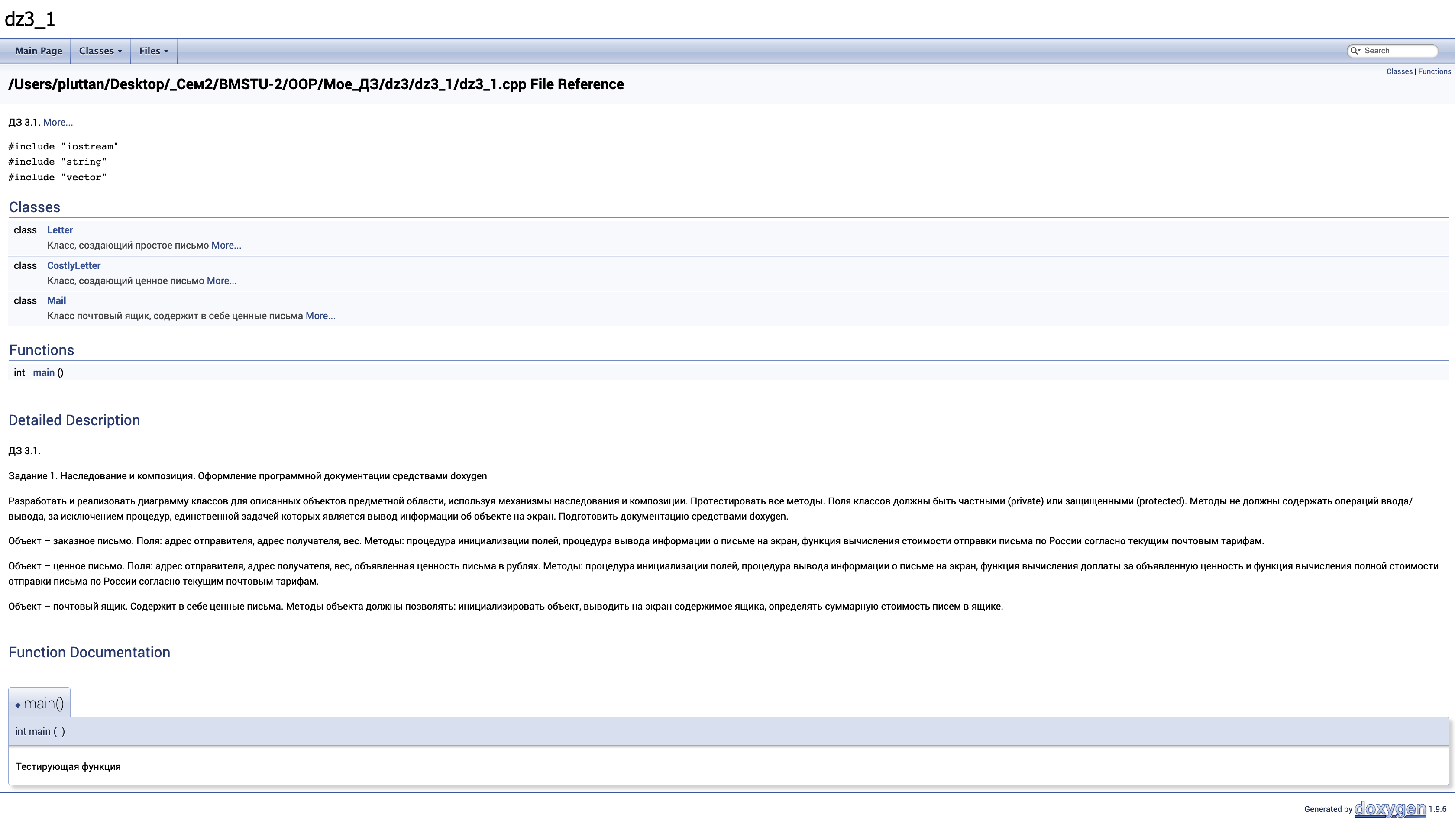
 











***Часть 3.2. Меню. Стандартные диалоги Qt***

Разработать программу, открывающую текстовый файл на редактирование (путь к файлу указывает пользователь), а также возможность сохранения измененного файла под новым именем. Предусмотреть возможность выбора шрифта отображения.

Notebook.h

#ifndef NOTEBOOK\_H

#define NOTEBOOK\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui { class notebook; }

QT\_END\_NAMESPACE

class notebook : public QMainWindow

{

Q\_OBJECT

public:

notebook(QWidget \*parent = nullptr);

~notebook();

private slots:

void on\_choose\_clicked();

void on\_save\_as\_clicked();

void on\_lineEdit\_textChanged(const QString &arg1);

void on\_fontComboBox\_currentFontChanged(const QFont &f);

void on\_save\_clicked();

private:

Ui::notebook \*ui;

};

#endif // NOTEBOOK\_H

Notebook.cpp

#include "notebook.h"

#include "ui\_notebook.h"

#include <QFileDialog>

#include <QDebug>

notebook::notebook(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::notebook)

{

ui->setupUi(this);

}

notebook::~notebook()

{

delete ui;

}

void notebook::on\_choose\_clicked()

{

ui->lineEdit->setText(QFileDialog::getOpenFileName(this, "Выберите файл", QDir::currentPath()));

}

void notebook::on\_save\_as\_clicked()

{

ui->lineEdit->setText(QFileDialog::getSaveFileName(this, "Сохранить файл как...", QDir::currentPath()));

on\_save\_clicked();

}

void notebook::on\_lineEdit\_textChanged(const QString &arg1)

{

QFile file(arg1);

if(file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QByteArray t = file.readAll();

ui->plainTextEdit->setPlainText(QString(t));

file.close();

}

}

void notebook::on\_fontComboBox\_currentFontChanged(const QFont &f)

{

ui->plainTextEdit->setFont(f);

}

void notebook::on\_save\_clicked()

{

QFile file(QString(ui->lineEdit->text()));

QByteArray a = ui->plainTextEdit->placeholderText().toLocal8Bit();

if (file.open(QIODevice::WriteOnly))

{

file.write(ui->plainTextEdit->toPlainText().toUtf8());

file.close();

}

}

Main.cpp

#include "notebook.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

notebook w;

w.show();

return a.exec();

}

Notebook.ui

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

<ui version="4.0">

<class>notebook</class>

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

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>826</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>notebook</string>

</property>

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

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

<item>

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

<item>

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

<property name="text">

<string>Введите название файла:</string>

</property>

</widget>

</item>

<item>

<spacer name="horizontalSpacer">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

</layout>

</item>

<item>

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

<item>

<widget class="QLineEdit" name="lineEdit">

<property name="minimumSize">

<size>

<width>400</width>

<height>0</height>

</size>

</property>

</widget>

</item>

<item>

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

<property name="maximumSize">

<size>

<width>100</width>

<height>16777215</height>

</size>

</property>

<property name="text">

<string>Выбрать...</string>

</property>

</widget>

</item>

<item>

<spacer name="horizontalSpacer\_2">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

</layout>

</item>

<item>

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

<item>

<widget class="QFontComboBox" name="fontComboBox"/>

</item>

<item>

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

<property name="text">

<string>Сохранить</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="save\_as">

<property name="text">

<string>Сохранить как...</string>

</property>

</widget>

</item>

</layout>

</item>

<item>

<widget class="QPlainTextEdit" name="plainTextEdit"/>

</item>

</layout>

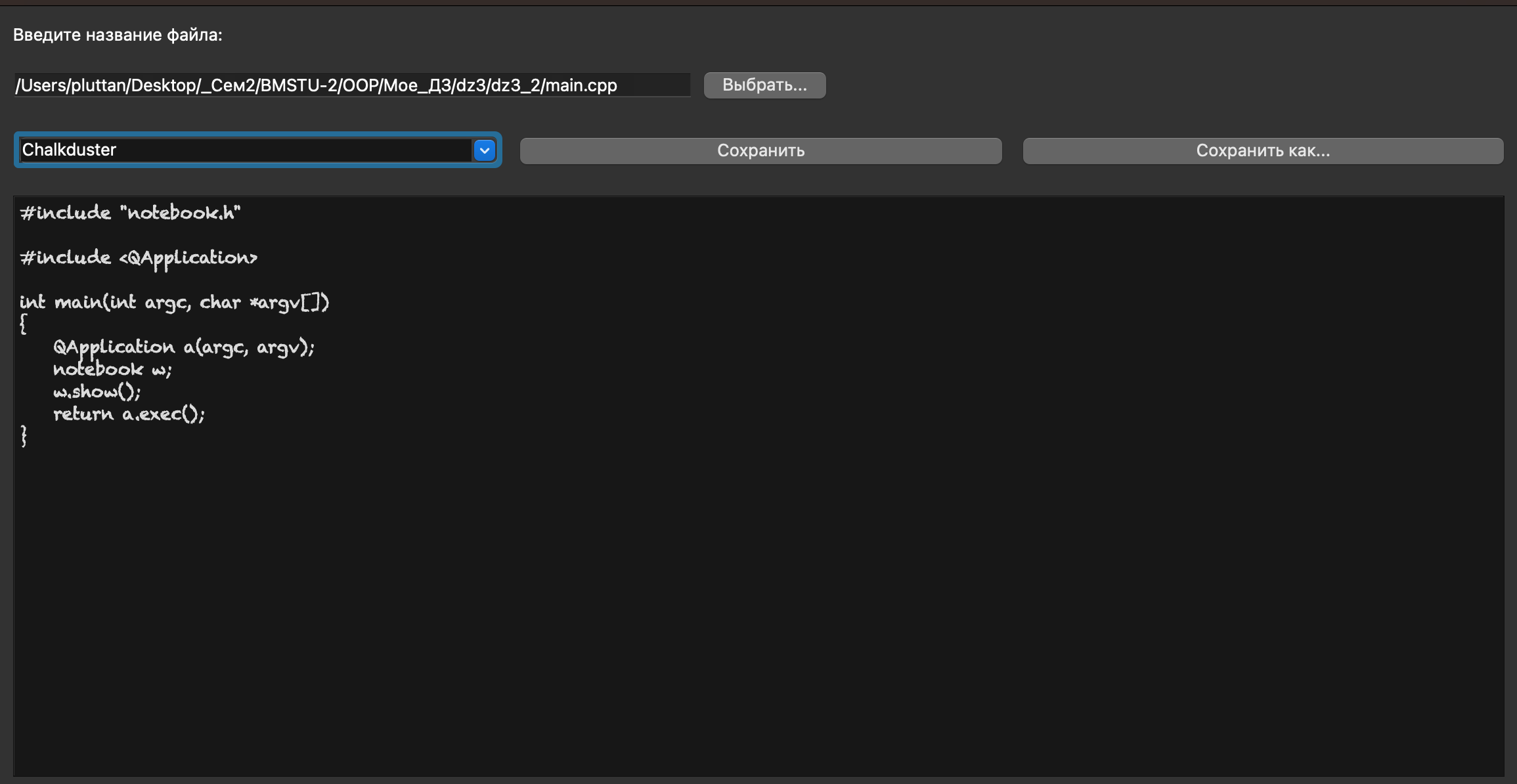
</widget>

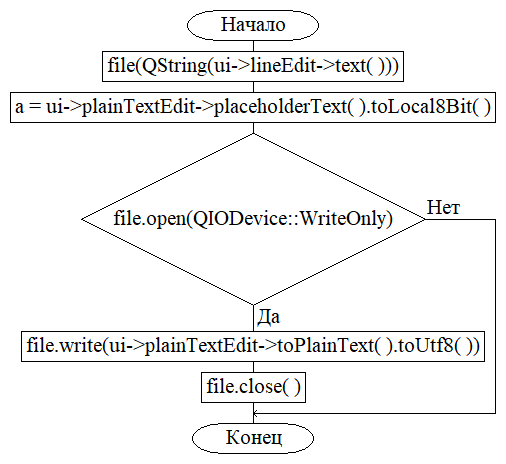
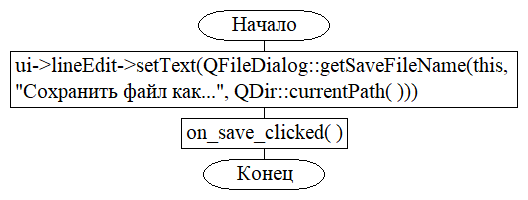
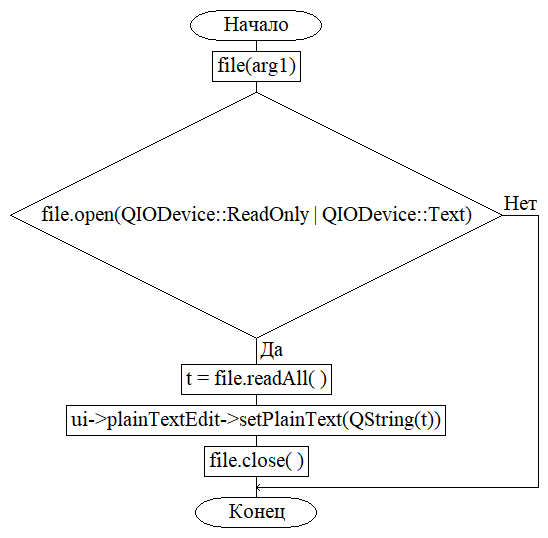
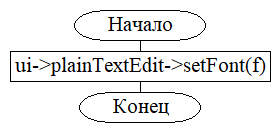
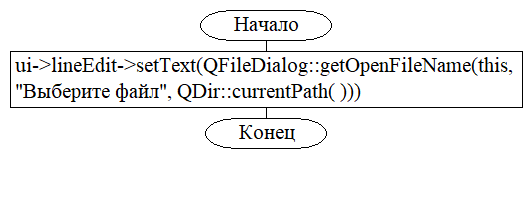
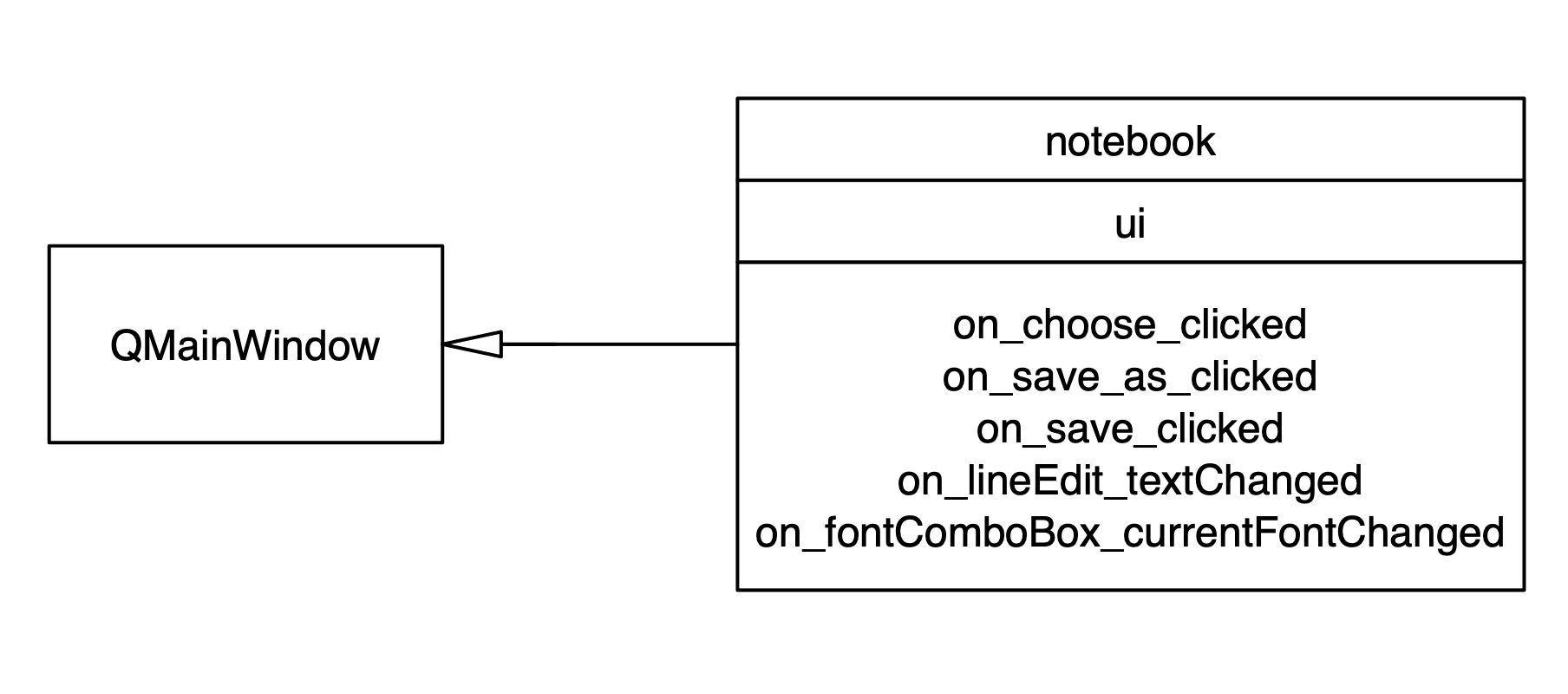
</widget>

<resources/>

<connections/>

</ui>



******

***Часть 3.3. Классы Model/View***

Разработать приложение, предназначенное для создания, редактирования и сохранения таблицы товаров с тремя колонками (название, цена, количество). Предусмотреть контроль вводимой информации в числовых полях.

Использовать средства Qt для реализации концепции Model/View.

Table.h

#ifndef TABLE\_H

#define TABLE\_H

#include <QMainWindow>

#include <QtSQL>

#include <QtDebug>

#include <QFileInfo>

#include <QSqlDatabase>

#include <QMessageBox>

QT\_BEGIN\_NAMESPACE

namespace Ui { class table; }

QT\_END\_NAMESPACE

class table : public QMainWindow

{

Q\_OBJECT

public:

table(QWidget \*parent = nullptr);

~table();

QSqlDatabase db;

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

private:

Ui::table \*ui;

};

#endif // TABLE\_H

Table.cpp

#include "table.h"

#include "ui\_table.h"

table::table(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::table)

{

ui->setupUi(this);

db = QSqlDatabase::addDatabase("QSQLITE");

db.setDatabaseName("/Users/pluttan/Desktop/\_Сем2/BMSTU-2/OOP/Мое\_ДЗ/dz3/dz3\_2/tableDB.db");

if (db.open()){

QMessageBox::information(this,"connection","connect");

}

}

table::~table()

{

delete ui;

}

void table::on\_pushButton\_clicked()

{

QSqlQuery que;

if (que.exec("CREATE TABLE IF NOT EXISTS " +

ui->lineEdit\_4->text() +

" (name varchar(50) PRIMARY KEY, price int, count int);")){

} else QMessageBox::information(this,"error","Something went wrong :(");

QSqlQueryModel \*modal = new QSqlQueryModel();

QSqlQuery\* q = new QSqlQuery(db);

q->prepare("select \* from " + ui->lineEdit\_4->text());

if(q->exec()){

modal->setQuery(\*q);

ui -> tableView -> setModel(modal);

}else QMessageBox::information(this,"error","Something went wrong :(");

}

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

{

QSqlQuery que;

QRegularExpression re("\\d+");

QRegularExpressionMatch match1 = re.match(ui->lineEdit\_2->text());

QRegularExpressionMatch match2 = re.match(ui->lineEdit\_3->text());

if (!(match1.hasMatch() and match2.hasMatch())) {

QMessageBox::information(this,"error","Type error :(");

return;

}

if (que.exec("REPLACE INTO "+ui->lineEdit\_4->text()+

" (name, price, count) VALUES ('"+ui->lineEdit->text()+

"', "+ui->lineEdit\_2->text()+", "+ui->lineEdit\_3->text()+

");")){

} else QMessageBox::information(this,"error","Something went wrong :(");

on\_pushButton\_clicked();

}

Main.cpp

#include "table.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

table w;

w.show();

return a.exec();

}

Table.ui

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

<ui version="4.0">

<class>table</class>

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

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>table</string>

</property>

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

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

<item>

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

<item>

<widget class="QLineEdit" name="lineEdit\_4">

<property name="minimumSize">

<size>

<width>0</width>

<height>30</height>

</size>

</property>

<property name="placeholderText">

<string>Название новой таблицы</string>

</property>

</widget>

</item>

<item>

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

<property name="minimumSize">

<size>

<width>0</width>

<height>30</height>

</size>

</property>

<property name="text">

<string>Создать новую таблицу/Найти существующую</string>

</property>

</widget>

</item>

</layout>

</item>

<item>

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

<item>

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

<item>

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item>

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

<property name="text">

<string>Название</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="lineEdit">

<property name="minimumSize">

<size>

<width>0</width>

<height>30</height>

</size>

</property>

<property name="placeholderText">

<string>Введите название</string>

</property>

</widget>

</item>

<item>

<widget class="QLabel" name="label\_2">

<property name="text">

<string>Цена</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="lineEdit\_2">

<property name="minimumSize">

<size>

<width>0</width>

<height>30</height>

</size>

</property>

<property name="placeholderText">

<string>Введите цену</string>

</property>

</widget>

</item>

<item>

<widget class="QLabel" name="label\_3">

<property name="text">

<string>Количество</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="lineEdit\_3">

<property name="minimumSize">

<size>

<width>0</width>

<height>30</height>

</size>

</property>

<property name="placeholderText">

<string>Введите количество</string>

</property>

</widget>

</item>

<item>

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

<property name="minimumSize">

<size>

<width>0</width>

<height>40</height>

</size>

</property>

<property name="text">

<string>Добавить / Изменить</string>

</property>

</widget>

</item>

<item>

<spacer name="verticalSpacer">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

</layout>

</item>

<item>

<widget class="QTableView" name="tableView">

<property name="minimumSize">

<size>

<width>400</width>

<height>0</height>

</size>

</property>

</widget>

</item>

</layout>

</item>

</layout>

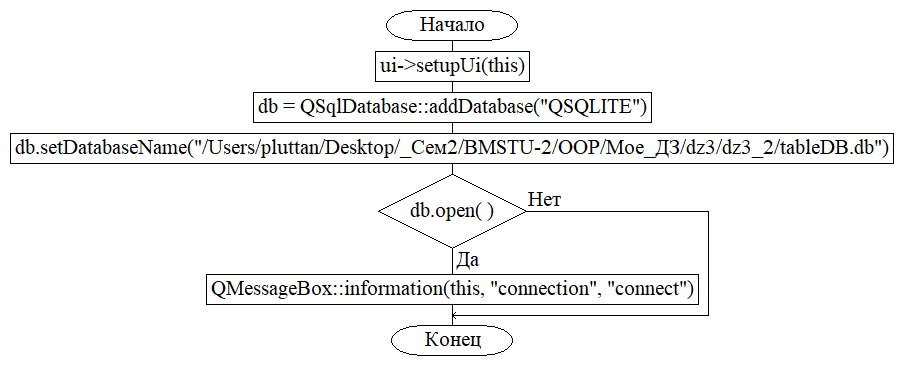
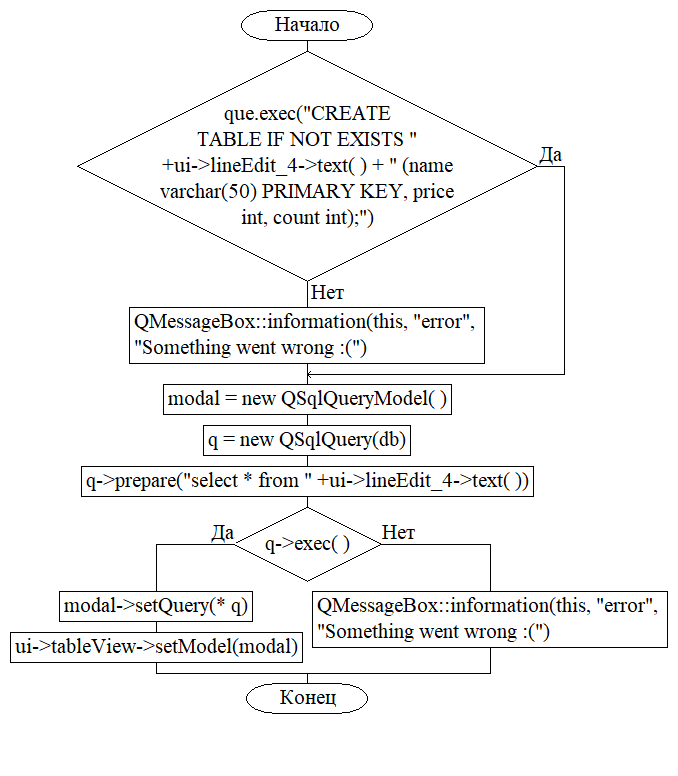
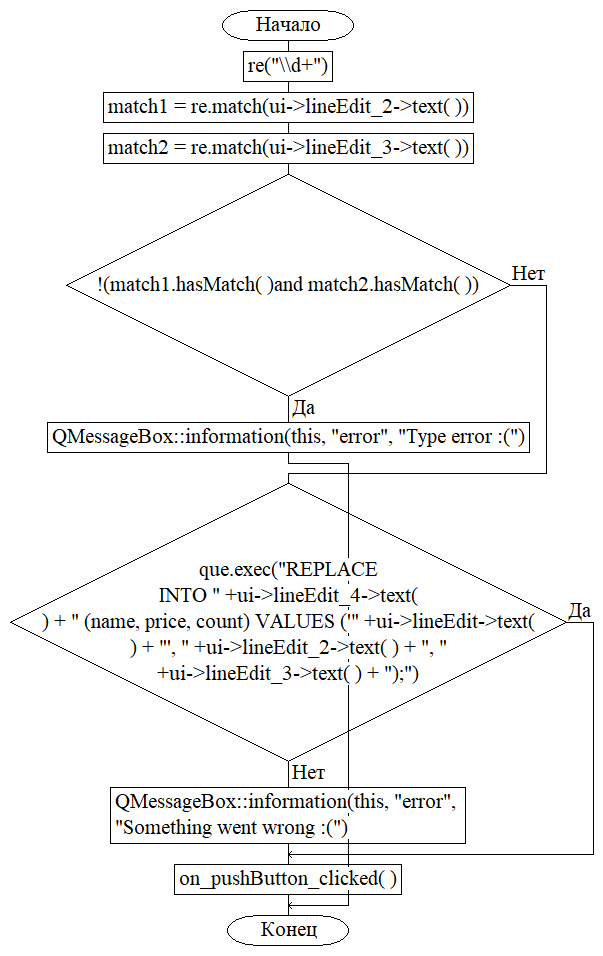
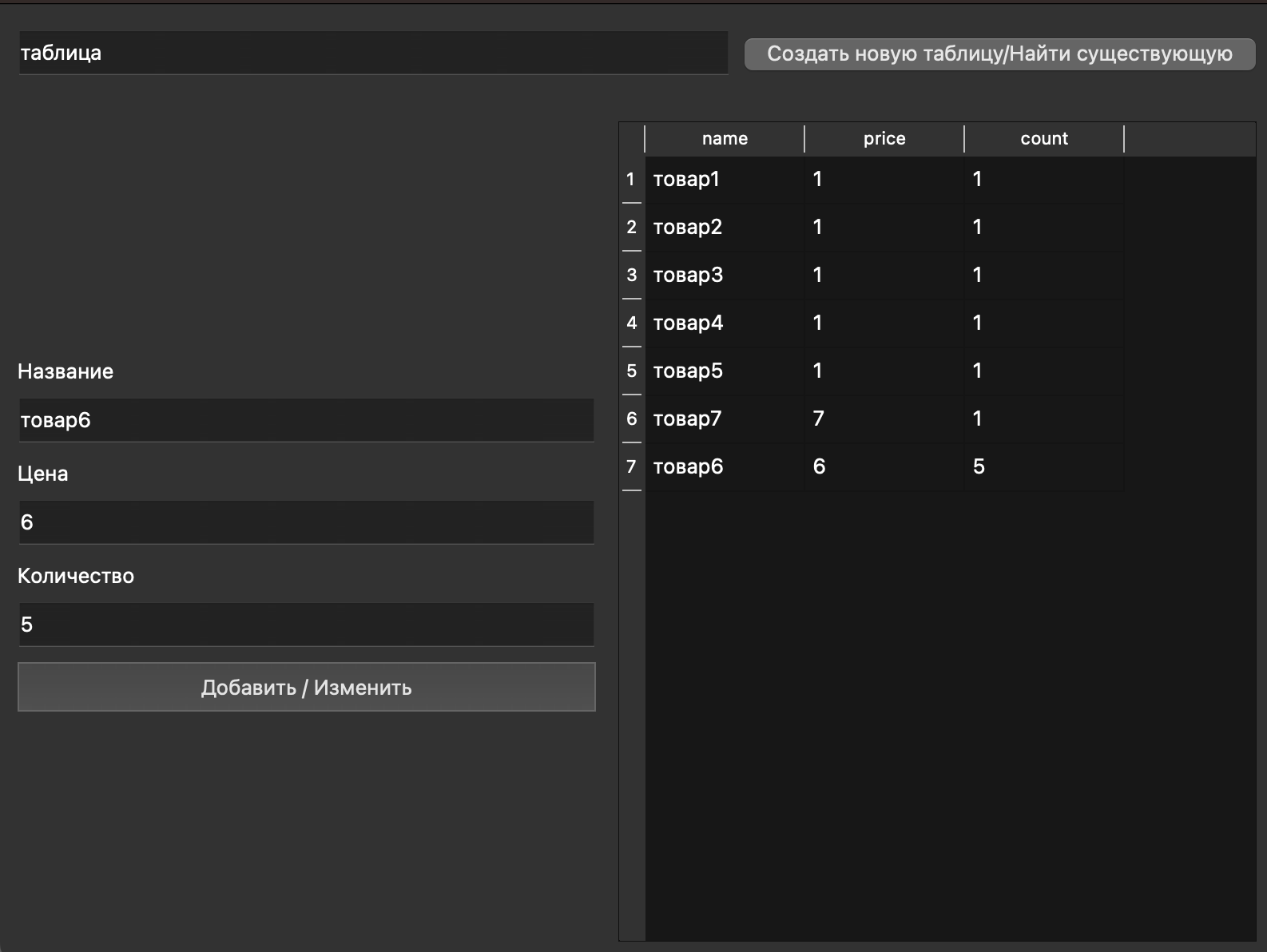
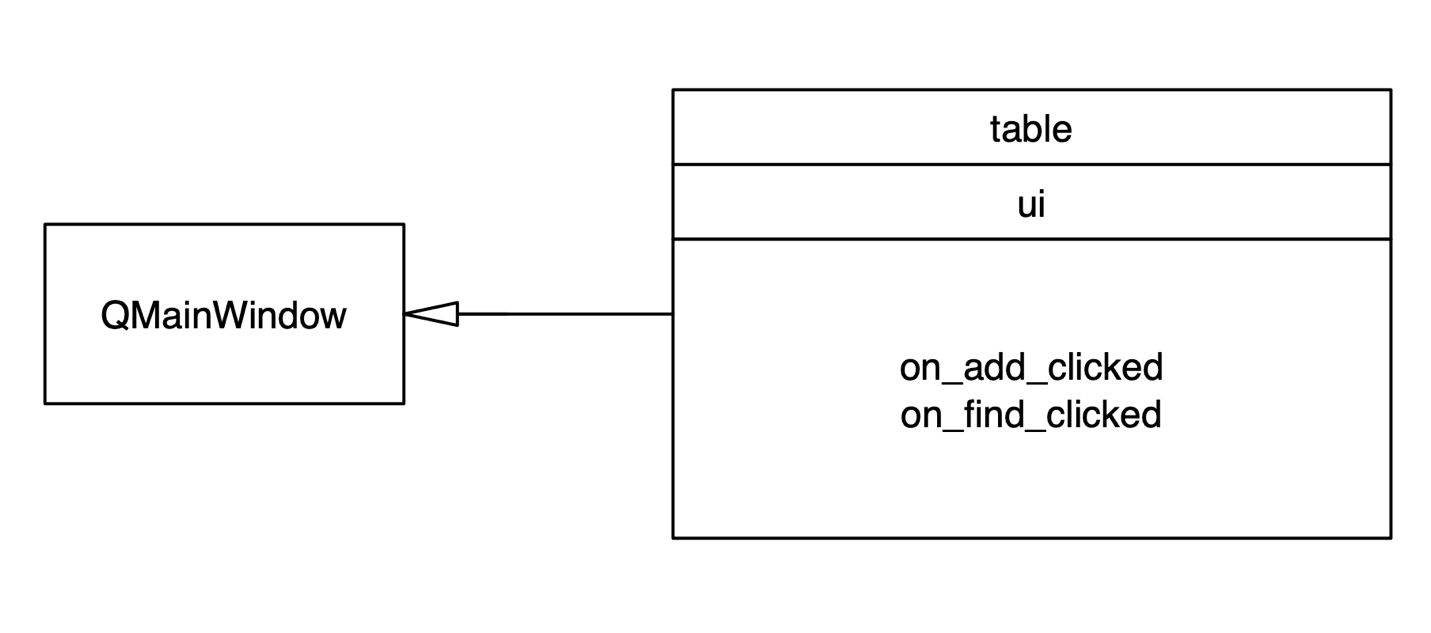
</widget>

</widget>

<resources/>

<connections/>

</ui>



Вывод: я научился работать со средствами qt, а так же составлять документацию.