Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ
Заведующий кафедрой
_____ В.В. Шайдуров
«20» июня 2016 г.

БАКАЛАВРСКАЯ РАБОТА

Направление 02.03.01 Математика и компьютерные науки

ЭЛЕКТРОННАЯ СИСТЕМА АДАПТИВНОГО ОБУЧЕНИЯ

Научный руководитель
кандидат технических наук,
доцент _____ Л.И. Покидышева
20.06.16

Выпускник _____ Я.В. Сизых
20.06.16

Красноярск 2016

РЕФЕРАТ

Выпускная квалификационная работа дипломная работа по теме «Электронная система адаптивного обучения» содержит 63 страницы текста и 9 использованных источника.

ПРОГРАММИРОВАНИЕ, ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ, ЭФФЕКТИВНОЕ ОБУЧЕНИЕ, АДАПТАЦИЯ, ТЕСТИРОВАНИЕ, БАЗЫ ДАННЫХ.

Цель работы – разработка и написание программы, представляющей собой электронную систему адаптивного обучения. Основная задача данной системы: на основе предварительных тестов выдавать обучаемому только тот материал, который ему неизвестен или известен в наименьшей степени.

В результате разработки была реализована программа со всем необходимым функционалом, а также средство для редактирования и создания новых учебных пособий.

Данная система может быть актуальна для тех, кто хочет в сжатые сроки изучить максимум информации. А также для тех, кто хочет изучить новый предмет с максимальной эффективностью.

Примером эффективного использования данной системы может явиться, например, подготовка к государственному экзамену, для успешной сдачи которого необходимы знания во многих областях науки и для подготовки к которому у студентов уходит очень много времени и сил из-за слишком большого объема информации. Важно отметить, что большая часть времени уходит на фильтрацию ненужной информации, которая студенту и так известна и на поиск того, что он мог забыть и это необходимо вспомнить. Данная система могла бы сильно повысить эффективность такой подготовки.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время активно развиваются системы дистанционного обучения и сейчас получение полноценного образования практически по любому предмету дистанционно уже не является проблемой. На данный момент существует огромное множество разнообразных систем дистанционного обучения (так называемых e-Learning Management System, или LMS), как онлайн, так и автономных. Самыми распространенными являются такие системы, как Moodle, ILIAS, aTutor и им подобные.

И хотя онлайн-обучение и имеет ряд преимуществ, такие как: обучение в индивидуальном темпе, свобода и доступность, но оно также имеет и один серьезный недостаток. Этот недостаток заключается в том, что ни одна система дистанционного обучения не подстраивается под обучаемого так, как это мог бы делать настоящий, живой преподаватель. Все эти системы просто выдают весь материал, глава за главой, которые заложили в них авторы, независимо от того, был ли уже известен этот материал студенту или нет. Поэтому довольно часто на прочтение больших, объемных текстов у студентов не хватает ни времени, ни желания.

В связи с этим, главная цель моей бакалаврской работы – это создание электронной системы обучения, которая бы гибко подстраивалась под каждого обучаемого индивидуально, была адаптивной и не нагружала бы студента лишней, ненужной или уже известной ему информацией. В некотором роде, эта система должна быть как живой преподаватель, который если видит, что студент уже знает материал, давал бы ему только новую информацию, которую ему необходимо усвоить, при этом в понятной и доступной форме.

Также важным моментом для эффективного обучения является проверка знаний студента после освоения каждой главы электронного учебника. Это позволит ему оценить то, насколько успешно он усвоил пройденный материал и, при необходимости, смог бы вернуться и повторить его.

1 ПОСТАНОВКА ЗАДАЧИ

Задача данной работы – написание программы, которая являлась бы электронной системой обучения и реализовать в ней следующие возможности:

1. Автономность – система должна быть способна работать на компьютере студента (далее – пользователя) без обязательного подключения к интернету и представлять собой исполняемый файл.

2. Адаптивность – перед каждой главой учебника, пользователь должен проходить небольшое тестирование, на основе результатах которого, система автоматически сформирует главу и включит в неё только необходимый пользователю материал. При этом необходимо также предоставить ему доступ и к полной версии учебника, если он не захочет проходить предварительное тестирование.

3. Итоговое тестирование пользователя – после каждой главы пользователь (по желанию) может пройти итоговое тестирование, в результате которого программа сообщит, насколько хорошо пользователь освоил материал и укажет на неверные ответы, чтобы он смог вернуться в учебник и еще раз повторить эти моменты.

4. Редактирование и создание нового учебника – в программе должна быть функция, позволяющая пользователю редактировать существующие и создавать новые учебники.

5. Хранение и загрузка учебников в облаке для удобного обмена учебными материалами между преподавателями и студентами.

2.2 РАЗРАБОТКА И НАПИСАНИЕ ПРОГРАММЫ

2.1 ВЫБОР ИНСТРУМЕНТОВ И ЯЗЫКА ПРОГРАММИРОВАНИЯ

Для разработки системы будем использовать интегрированную среду разработки (так называемую IDE) MS Visual Studio Community 2015. Эта IDE обладает достаточным для написания нашей программы функционалом, она актуальна, при этом бесплатна и имеет большое количество расширений, помогающих в написании программы.

Языком программирования был выбран C#, как наиболее эффективный, актуальный и при этом весьма гибкий язык программирования. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов). Он сочетает в себе достоинства управляемого (managed) кода .NET (безопасность типов, сборка мусора, структурная обработка исключений и т.д.)., и при этом позволяет взаимодействовать с неуправляемым (unmanaged) кодом (например C++) (достоинства: скорость выполнения, гибкость) представленного в виде библиотек DLL.

Помимо среды разработки и языка программирования необходимо также выбрать способ организации хранения учебных материалов и всех служебных данных. Очевидно, что учебник необходимо хранить отдельно от самой программы. Для этих целей было принято решение использовать реляционную базу данных. В качестве системы управления базой данных (СУБД) будем использовать популярную встраиваемую СУБД SQLite. Слово «встраиваемый» (embedded) в данном случае означает, что SQLite не использует парадигму клиент-сервер, то есть библиотека SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и библиотека становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном

стандартном файле на том компьютере, на котором исполняется программа. При этом исходный код SQLite лежит в открытом доступе и является общественным достоянием, что позволяет нам свободно её использовать.

Исходный текст самого учебника будет написан на языке гипертекстовой разметки HTML (HyperText Markup Language). HTML – очень гибкий и удобный инструмент для создания и разметки текстовых файлов, с возможностью вставки мультимедийного содержания практически любого формата (картинки, аудио и видеофайлы), что как раз подходит для любого вида учебного материала. Таким образом, каждая глава учебника будет представлять собой небольшой сайт, который будет отображаться в окне программы.

2.2 ПЛАНИРОВАНИЕ

2.2.1 РАЗРАБОТКА СХЕМЫ ИНТЕРФЕЙСА

Для начала необходимо схематично представить, как будет выглядеть интерфейс нашей будущей программы.

При запуске, программа первым делом должна отобразить пользователю главное окно программы, в котором должны быть доступны операции открытия учебника, загрузка учебника онлайн, а также открытие редактора.

Таким образом, наглядно схема интерфейса программы будет выглядеть примерно следующим образом:

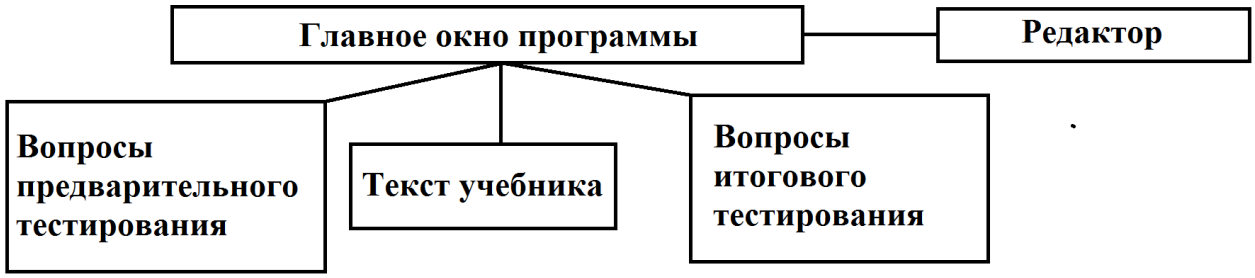

Рисунок 2.1 – Схема интерфейса программы

Если пользователь выбирает «Открыть в редакторе», то программа откроет загруженный в данный момент учебник в окне с формой редактирования учебника со всеми необходимыми для этого функциями. При этом редактор должен быть доступен, как самостоятельная программа, отдельно от самого учебника.

Если пользователь выбирать открытие учебника, то программа показывает диалогое окно загрузки учебника, где пользователь может открыть нужный ему учебник. После открытия учебника в главном окне пользователю на выбор должно предоставляться: прохождение предварительного тестирования, открытие полнуй версии учебника без тестирования, переход к финальному тестированию.

При открытии полной версии учебника, программа загружает весь текст учебника независимо от результатов и отображает пользователю весь

доступный материал. При желании, здесь же пользователь может вернуться к предварительному или же финальному тестированию.

При выборе итогового тестирования, программа отображает все вопросы итогового тестирования, где пользователь может проверить свои знания. Отсюда пользователь может вернуться к сгенерированному для него учебнику и повторить необходимый материал.

Если же пользователь выбирает вариант вступительного тестирования, то программа отобразит форму предварительного тестирования. После заполнения формы, пользователь может перейти к тексту учебника, где будет сформирован материал. Также он может перейти к сразу к финальному тестированию

Отметим также, что из главного окна программы должна быть реализована возможность открытия окна параметров программы, а также окна со справкой.

2.2.2 ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММЫ

Для эффективного написания программы, прежде всего необходимо примерно описать структуру программы: из каких модулей она будет состоять и взаимосвязь между этими модулями.

Но для этого сначала определимся с методологией программирования. Для подобного рода программы наиболее подходящим будет подход структурного программирования. В программе предполагается наличие весьма небольшого числа отдельных объектов, так что нет никакой необходимости использовать объектно-ориентированное программирование, наличие классов лишь усложнит структуру без надобности.

В соответствии с методологией структурного программирования, любая программа строится без использования оператора goto из трёх базовых управляющих структур:

1. Последовательность;

2. Ветвление;

3. Цикл.

Кроме того, используются подпрограммы. При этом разработка программы ведётся пошагово, методом «нисходящего программирования».

Итак, структурной единицей нашей программы будет форма окна, весь код будет содержаться в них.

Итого у нас будет всего пять форм:

1. Основная форма учебника;

2. Форма настроек;

3. Форма загрузки учебников онлайн;

4. Форма создания/редактирования учебника.

Все окна, кроме редактора учебника, будут создаваться из первой формы, все необходимые данные будут передаваться также из главной формы.

Помимо форм, также будет реализован класс “BDClass” для удобной работы с базой данных и загрузкой учебника.

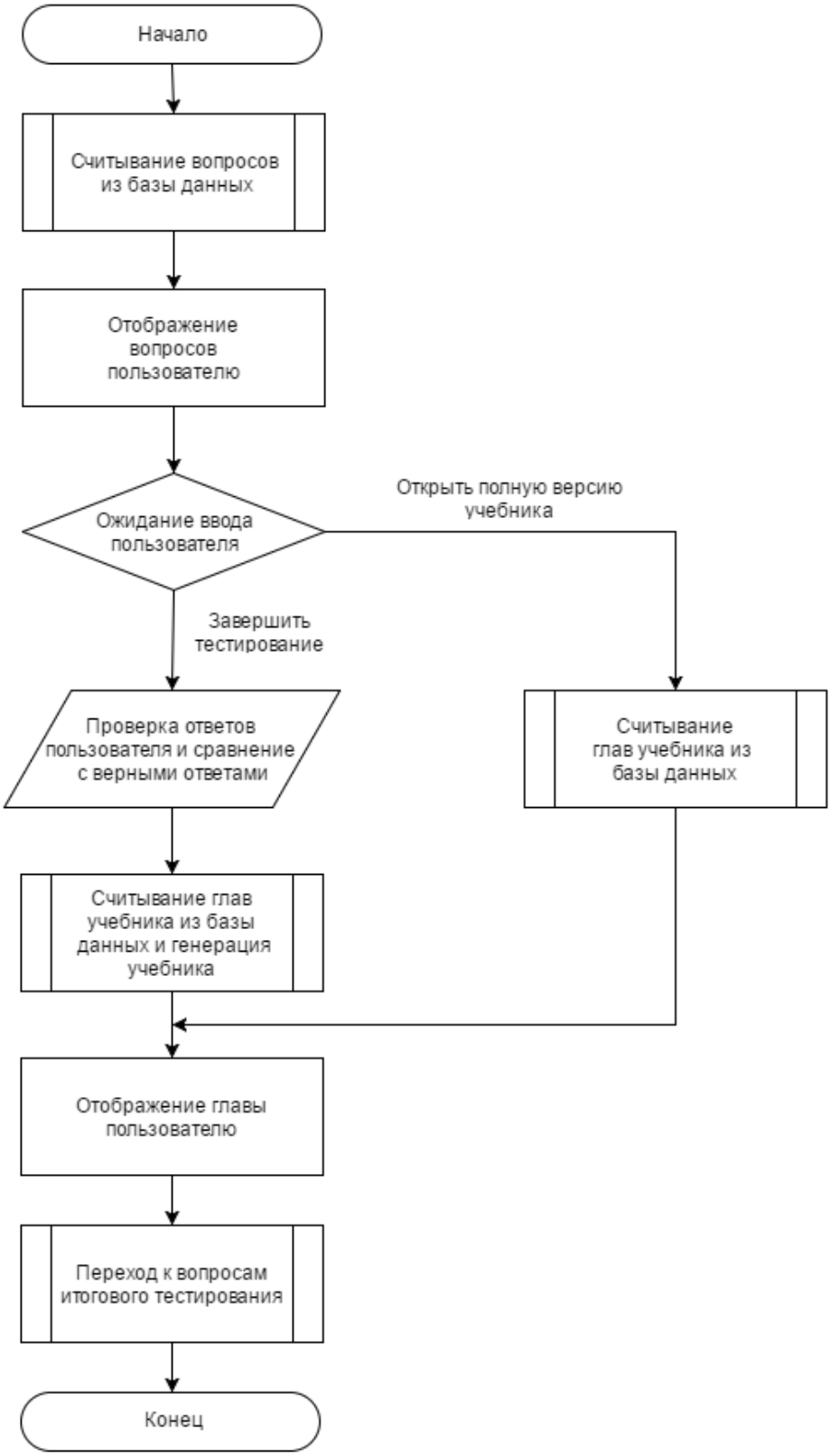

Рисунок 2.2 – Блок-схема работы алгоритма предварительного тестирования.

2.2.3 ОПИСАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ И ФАЙЛА С ДАННЫМИ УЧЕБНИКА

Прежде, чем переходить к написанию самой программы, нужно еще определить, каким образом служебные данные и данные учебника будут храниться в нашей базе данных. Здесь же опишем, каким образом всё это будет между собой взаимосвязано.

Для редактирования баз данных SQLite существует множество программ, например, открытая и свободная sqliteman или sqliteadmin.

Один учебник можеть храниться в одной базе данных. Данные о главах будут храниться в отдельной таблице “books_list”. Данные вступительного, итогового тестирования и параграфов учебника будут храниться в таблицах “start_quest”, “final_test” и “paragraphs” соответственно.

Таблица со списком глав “books_list” состоит из трех полей:

1. id – уникальное числовое значение, обозначающее идентификационный номер каждой главы. По этому номеру из таблиц с данными будут выбираться соответствующие данной главе данные.
2. name – текстовая строка, обозначающая название главы учебника.
3. Descritption – текстовое поле с описанием данного учебника. Данное описание отображается при загрузке каждой главы, является начальной страницей с вводным текстом (допускается использование HTML-тэгов).

Каждая таблица с данными учебника состоит из 5 основных полей:

1. text – поле текстового типа. В этом поле содержится либо текст вопроса, либо содержание части учебника (допускается использование HTML-тэгов).

2. points – числовое значение, обозначающее количество баллов. В случае, если это поле с вопросом, то оно означает количество баллов, которое дается за верный ответ. В случае, если это глава учебника, то оно означает, что если пользователь, отвечая на вопросы предварительного тестирование, набрал меньше баллов, чем указано в points, то данная глава учебника будет ему отображена. Если же он набрал больше, то глава выводиться не будет, т.к. подразумевается, что пользователю информация в этой главе уже известна.

3. name – текстовая строка, обозначающая внутреннее название либо вопроса, либо главы учебника. Непосредственно в учебнике не отображается, выводится только в редакторе, для удобства.

4. id – числовое значение, обозначает к какой главе относится данный элемент.

5. index – числовое значение, обозначающее порядок, в котором элементы будут отображаться пользователю.

Также у таблиц начального тестирования и параграфов есть следующая строка:

key – это текстовая строка, означающая связь между какой-либо главой учебника и каким-либо вопросом предварительного тестирования. Суть в следующем: если пользователь неверно отвечает на какой-то вопрос, ключевое значение которого совпадает с ключевым значением какой-либо главы учебника, то эта глава ему будет обязательно показана, независимо количества набранных баллов. И наоборот, если он ответил верно, то глава показана не будет. Количество связанных вопросов и глав неограниченно.

У таблиц предварительного и итогового тестирования также есть строка с ответами:

answer – текстовое поле. В этом поле содержатся верные ответы на вопросы предварительного или итогового тестирования. Отметим, что вопросы могут быть двух типов: открытый (ответ записывается свободно в виде текстовой строки), закрытый вопрос в четырьмя вариантами ответов. В случае вопроса с четырьмя вариантами, answer содержит просто цифру с порядковым номером верного ответа (от 1 до 4).

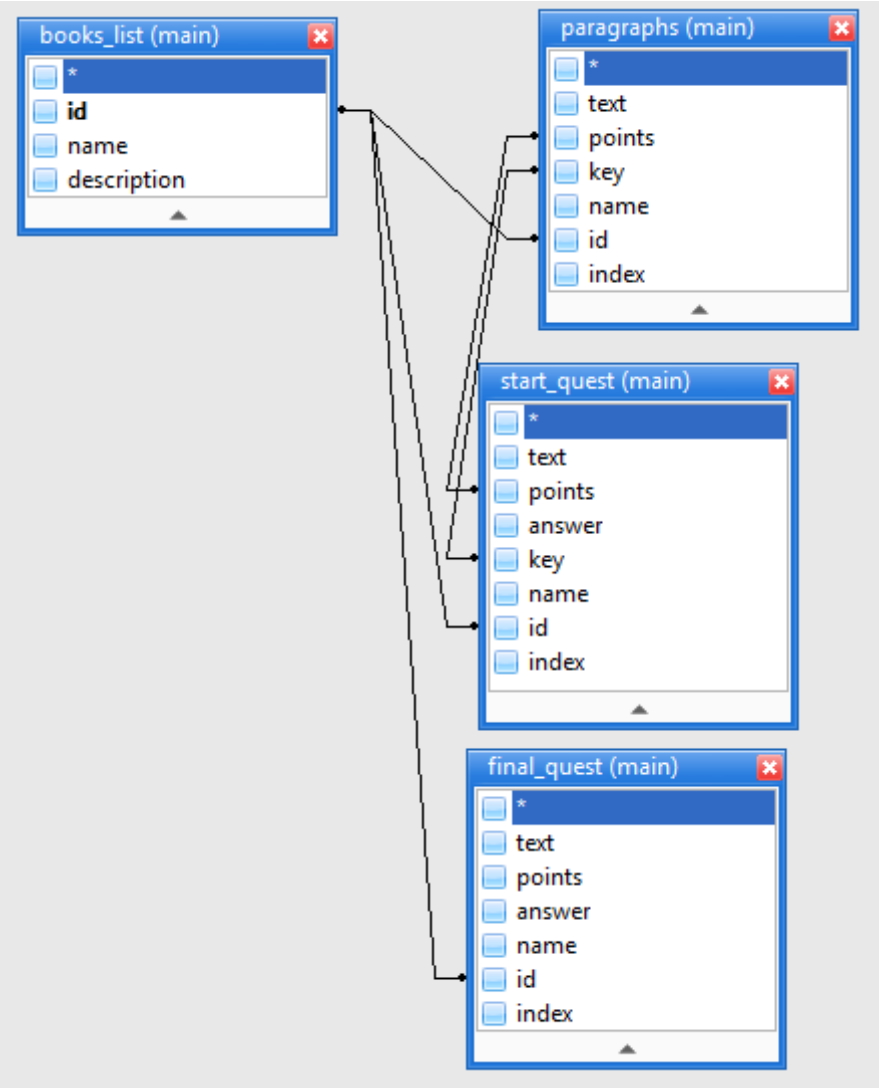

Рисунок 2.3 – Схема базы данных

База данных учебника имеет название “base.db” и содержится в файле учебника. Файл учебника представляет собой обычный ZIP-архив с расширение *.stb (SmartTextBook). Помимо базы данных содержит в себе папку “pic” с файлами изображений учебника.

2.3 НАПИСАНИЕ КОДА

Итак, приведем процесс написания кода нашей программы. Из-за большого объема, весь код приведен не будет, только основные его части.

Функции и классы, используемые в программе учебника SmartTextBook:

BDClass.cs – класс, который загружает файл учебника и открывает базу данных. Имеет следующие методы: Set и UnSet, отвечающие соответственно, за загрузку и выгрузку файла учебника.

STBForm.cs – класс формы основного окна программы. Содержит в себе все основные функции для работы с учебником – загрузка и отображение вопросов, глав учебника, проведение тестирования и проверка результатов и, соответственно, выводит на экран основное окно программы. Имеет следующие методы:

STBForm_Load – функция загрузки формы окна. Загружает форму и выводит приветствие.

LoadListBooks – функция загрузки списка всех глав учебника.

LoadBook – функция загрузки учебника.

OpenSQ – функция загрузки и открытия вопросов предварительного тестирования

button5_Click – функция, запускающаяся при нажатии кнопки «Завершить тестирование». Проверяет ответы пользователя и выводит результат.

OpenParagraphs – функция загрузки и открытия глав учебника.

OpenFQ – функция загрузки и открытия вопросов итогового тестирования.

LoadOnlineForm.cs – класс формы окна загрузки учебника с онлайн-сервера. Основные методы класса:

LoadOnlineForm_Load – функция загруки формы. Отображает окно формы.

LoadServersList – функция загрузки списка серверов.

LoadBooksList – функция загрузки списка книг с серверов.

button1_Click – функция, запускающаяся при нажатии кнопки «Добавить». Добавляет в список серверов новый адрес сервера.

button5_Click – аналогичная функция для удаления адреса сервера.

button4_Click – функция кнопки загруки списка учебников.

button3_Click – функция кнопки загрузки учебника.

button6_Click – функция открытия загруженного учебника.

EditorForm.cs – класс формы окна редактора учебников. Методы данного класса:

Form1_Load – функция загрузки окна формы. Отображает окно с приветствием.

CreateBook – функция создания нового учебника. Создает новый файл учебника и сохраняет его в указанном месте.

button6_Click – функция кнопки добавления новой главы учебника.

LoadListElements – функция загрузки списка элементов учебника.

button1_Click – функция добавления нового элемента учебника.

textBox1_Leave – функция, выполняющаяся при покидании поля ввода. Сохраняет текст, введенный пользователем в базу данных.

button3_Click – функция кнопки отображения HTML-кода.

QuestionAdd.cs – класс формы окна добавления вопроса. Содержит методы:

QuestionAdd_Load – загрузка формы окна.

button3_Click – функция кнопки вставки нового изображения.

StylesForm.cs – класс формы окна редактора стилей. Содержит методы:

StylesForm_Load – загрузка формы окна.

comboBox1_SelectedIndexChanged – функция, выполняющаяся при изменении выбранного индекса выпадающего списка с категориями учебника.

button3_Click – функция кнопки, изменяющая фоновый цвет учебника.

2.3.1 КЛАСС ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ И ЗАГРУЗКИ УЧЕБНИКА (BDClass.cs)

Подключаем необходимые библиотеки: для работы с СУБД SQLite, для сохранения и загрузки настроек, а также для работы с потоками файлов (в остальных формах будут аналогичные строки):

```
using System.Data.SQLite;
using System.Data.Common;
using System.Configuration;
using System.IO;
```

Рассмотрим код конструктора класса:

Функция BDClass – перегруженный конструктор класса BDClass. В качестве параметра принимает строку filepath – путь к файлу. Эта функция загружает учебник и подключает базу данных:

```
       string zipPath = filepath;
            extractPath = System.IO.Path.GetTempPath() +
"SmartTextBook";
            if (config.AppSettings.Settings["WorkDir"] != null)
            {
                extractPath =
config.AppSettings.Settings["WorkDir"].Value;
            }
            if (Directory.Exists(extractPath))
            {
                try Directory.Delete(extractPath, true);
                catch { }
            }

            String file = extractPath + "\\base.db";

            ZipFile.ExtractToDirectory(zipPath, extractPath);


   SQL = "Data Source=" + file + "; Version=3;Journal Mode=Off";
            conn = new SQLiteConnection(SQL);
            try
            {
                conn.Open();
            }
            catch (SQLiteException ex)
            {
          MessageBox.Show(ex.Message, "Ошибка открытия учебника",
          MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            OpenedFile = filepath;
```

В качестве параметра принимается текстовая строка, в которой содержится полный путь до учебника. Мы создаём временную папку (это либо временная папка, либо же получаем её из файла с настройками) и в неё распаковываем с помощью функции ZipFile.ExtractToDirectory файлы нашего учебника.

Далее мы создаём SQL подключение для загрузки базы данный и открываем нашу базу данных.

В классе также имеется код загрузки учебника по запросу:

```
public void Set (String filepath) { ... }
```

Код рассмотрен не будет, он почти аналогичен предыдущему, за исключением некоторых моментов.

Рассмотрим код функций, отключающий базу данных, а также код функции проверяющий, загружен ли в данный момент учебник.

Функция UnSet. Отключает базу данных и очищает временную папку:

```
    conn.Close();
    SQLiteConnection.ClearAllPools();
    GC.Collect();
    GC.WaitForPendingFinalizers();
    SQL = "";
    try
    {
        if (Directory.Exists(extractPath)) Directory.Delete(extractPath, true);
    }
    catch { }
}
public bool IsSet()
{
    if ( SQL=="" ) return false;
    else return true;
```

В первой функции мы закрываем базу данных ждем, когда программа полностью освободит файл, очищаем SQL-запрос и после чего мы удаляем временную папку.

Во второй функции мы проверяем, установлен ли SQL-запрос. Если строка запроса пуста, то учебник не загружен и возвращается false, иначе true.

2.3.2 ПРОГРАММА УЧЕБНИКА (SmartTextBook.exe)

2.3.2.1 ФОРМА ОСНОВНОГО ОКНА (STBForm.cs)

Форма имеет перегруженный конструктор класса, отличающийся тем, что принимает в качестве аргумента элемент класса BDClass, с загруженным учебником.

Функция STBForm – перегруженный конструктор формы STB_Form. Принимает в качестве параметра элемент класса BDClass с загруженным учебником:

```
BD = file;
InitializeComponent();
```

Файл BD – это переменная типа BDClass, хранящая в себе данные о загруженном учебнике.

Рассмотрим код, выполняющийся при загрузке формы.

Функция STB_FormLoad – отображает главное окно программы и выводит приветствие:

```
        private void STBForm_Load(object sender, EventArgs e)
        {
            if (config.AppSettings.Settings["WorkDir"] != null)
            {
                CurrentPath = config.AppSettings.Settings["WorkDir"].Value;
            }
            pictureBox1.SendToBack();
            SetView("main");
            if (BD.IsSet())
            {
                LoadListBooks();
            }
            else
            {
                webBrowser1.DocumentText =  "<style> body {background-
color:#CAEBF3;color:#555;font-family:Arial;font-size:24px;}</style>" +
"<body><div width = \"60%\" align = \"center\"><h1>Добро пожаловать!</h1><br><strong>" +
 "Вы запустили Систему Адаптивного Обучения SmartTextBook.<br>" +
 "Для начала обучения откройте учебник:<br>Файл - Открыть учебник.<br>" +
 "<br>Вы также можете загрузить учебник с онлайн-облака: Файл - Загрузить учебник
онлайн.</strong></div><br>" +
 "<div width = \"60%\" align = \"center\"> Система создана в качестве выпускной
квалификационной работы "
+ "студентом СФУ Института Математики и Фундаментальной Информатики, <br> Сизых Я.В. Группа
ИМ12-06Б.<br><br><hr>2016г.</div></body>";
             if (comboBox1.Items.Count!=0 )comboBox1.SelectedIndex = 0;
            }        }
```

Здесь происходит считывание настроек, установка вида формы в режим «основной» (в функции SetView проиходит активация элементов формы), далее мы проверяем, загружен ли учебник и есть загружен, то загружаем его, иначе в окно браузера выводим приветственный текст.

Рассмотрим подробнее функцию загрузки списка глав учебника LoadListBooks.

Функция LoadListBooks – загружает список всех глав учебника:

```
            TestPass = false;
            comboBox1.Items.Clear();
            SQLiteCommand cmd = new SQLiteCommand(BD.conn);
            SQLiteDataReader r;
            Descriptions = new List<string>();
            IDs = new List<int>();

            cmd.CommandText = "SELECT * FROM [books_list]";
            try
            {
                r = cmd.ExecuteReader();
                int j = 0;
                foreach (DbDataRecord record in r)
                {
                    comboBox1.Items.Add("");
                    Descriptions.Add("");
                    IDs.Add(0);

                    comboBox1.Items[j] = record["name"].ToString();
                    Descriptions[j] = record["description"].ToString();
                    IDs[j] = Convert.ToInt32(record["id"]);
                    j++;
                }
            }
            catch (SQLiteException ex)
            {
                MessageBox.Show("Неверный файл учебника!\n\nОшибка:\n" + ex.Message,
"Ошибка открытия учебника", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            cmd.Dispose();

            if (comboBox1.Items.Count != 0)
            {
                comboBox1.SelectedIndex = 0;
                string text = Descriptions[comboBox1.SelectedIndex];
                text = text.Replace("'", "\"");
                text = text.Replace("src=\"", "src=\"" + CurrentPath + "\\pic\\");
                webBrowser1.DocumentText = text;
            }
```

Сначала мы присваиваем переменной TestPass значение false – это значит, что пользователь еще не прошел вступительное тестирование, далее мы подключаем базу данных с помощью нашей переменной BD, загружаем из неё список учебника их в элементе comboBox1. Если в списке будет хоть один элемент, то выбирается самый первый и его описание отображается в браузере.

При открытии окна, без действий пользователя на данный момент никаких событий происходить не будет. Рассмотрим, что будет, если пользователь захочет открыть новый учебник через меню Файл – Открыть учебник.

Функция открытьУчебникToolStripMenuItem_Click – вызывается при выбора пункта меню «Файл – Открыть учебник». Загружает выбранный учебник:

```
OpenFileDialog openFileDialog1 = new OpenFileDialog();
            openFileDialog1.InitialDirectory = Application.StartupPath;
            openFileDialog1.Filter = "Файлы SmartTextBook (*.stb)|*.stb";
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
                LoadBook(openFileDialog1.FileName);
```

В данной функции мы открываем диалоговое окно Windows с предложением выбора файла. Если пользователь выбрал какой-то файл, то мы загружаем его с помощью функции LoadBook. Рассмотрим теперь эту функцию. Функция LoadBook. Принимает в качестве аргумента строку file, содержащую адрес файла учебника. Загружает учебник с помощью методов класса BDClass:

```
if (BD.IsSet())
{
    try  {
        BD.UnSet();
    }
    catch (SQLiteException ex) {
        MessageBox.Show(ex.Message, "Ошибка открытия учебника",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }            }
try {
    BD.Set(file);

    if (BD.IsSet())
    {
        LoadListBooks();
    }
    CurrentFile = file;
    this.Text = file + " - SmartTextBook";
}
catch (SQLiteException ex)         {
    MessageBox.Show(ex.Message, "Ошибка открытия учебника",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}

if (BD.IsSet())      {
    LoadListBooks();
```

Здесь мы сначала проверяем, загружен ли учебник и если он загружен, то освобождаем его и очищаем базу данных с помощью метова нашего класса BDClass UnSet, далее с помощью конструкции try – catch пытаемся загрузить новый учебник с помощью метода BD.Set и если учебник загружается, то

загружается список глав с помощью функции LoadListBooks, иначе выводится сообщение об ошибке.

Теперь опишем функцию, которая будет вызываться при нажатии кнопки прохождения предварительного тестирования.

Функция button1_Click – функция кнопки открытия вопросов предварительного тестирования:

```
        if (!BD.IsSet()) MessageBox.Show("Откройте учебник!", "Ошибка: учебник не
открыт", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        else    {
            OpenSQ();
        }
```

Здесь мы проверяем, загружен ли учебник, и если да, то выполняем функцию OpenSQ, иначе выводим сообщение об ошибке. Функция OpenSQ.

Функция OpenSQ. Загружает и отображает список вопросов предварительного тестирования:

```
            SetView("start_quest");

            if (config.AppSettings.Settings["WorkDir"] != null)
            {
                CurrentPath = config.AppSettings.Settings["WorkDir"].Value;
            }

            SQLiteCommand cmd = new SQLiteCommand(BD.conn);
            SQLiteDataReader r;
            cmd.CommandText = "SELECT * FROM [start_quest] WHERE id=" +
IDs[comboBox1.SelectedIndex] + " ORDER BY CASE WHEN [index] IS NULL THEN 1 ELSE 0 END,
[index]";
            try
            {
                r = cmd.ExecuteReader();
                SQ_sumpoints = 0;
                SQ_questions = new List<string>();
                SQ_answers = new List<string>();
                SQ_keys = new List<string>();
                SQ_points = new List<double>();

                int j = 0;
                foreach (DbDataRecord record in r)
                {
                    SQ_questions.Add("");
                    SQ_answers.Add("");
                    SQ_keys.Add("");
                    SQ_points.Add(0);

                    SQ_questions[j] = r["text"].ToString();
                    SQ_answers[j] = r["answer"].ToString();
                    SQ_keys[j] = r["key"].ToString();
                    SQ_points[j] = Convert.ToDouble(r["points"]);
                    SQ_sumpoints += Convert.ToDouble(r["points"]);
                    j++;
                }
                SQ_count = j;
```

```
            }
            catch (SQLiteException ex)
            {
                MessageBox.Show("Неверный файл учебника!\n\nОшибка:\n" + ex.Message,
"Ошибка открытия учебника", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            cmd.Dispose();

            // загрузка стилей
            SQLiteCommand cmd_styles = new SQLiteCommand(BD.conn);
            SQLiteDataReader r_styles;
            String SQ_head = "", SQ_end = "";
            cmd_styles.CommandText = "SELECT * FROM [start_quest] WHERE id='style_head'
OR id='style_end'";
            try
            {
                r_styles = cmd_styles.ExecuteReader();
                foreach (DbDataRecord record in r_styles)
                {
                    if (r_styles["name"].ToString() == "head") SQ_head +=
r_styles["text"].ToString();
                    if (r_styles["name"].ToString() == "end") SQ_end +=
r_styles["text"].ToString();
                }
            }
            catch (SQLiteException ex)
            {
                MessageBox.Show("Неверный файл учебника!\n\nОшибка:\n" + ex.Message,
"Ошибка открытия учебника", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            cmd_styles.Dispose();

            String temp = SQ_head;
            foreach (string s in SQ_questions)
            {
                temp += s;
            }
            temp += SQ_end;
            temp = temp.Replace("'", "\"");
            temp = temp.Replace("src=\"", "src=\"" + CurrentPath + "\\pic\\");
            webBrowser1.DocumentText = temp;
```

Здесь мы сначала устанавлием вид в режим предварительного тестирования (“start_quest”), далее загружаем настройки и далее подключаем базу данных из которой по выбранному ID загружаем список всех вопросов предварительного тестирования отсортированных по полю “index”. Помещаем их в предварительно объявленные переменные типа List.

После этого мы загружаем из базы данных две строки со стилями – head и end. Эти строки будут выводиться в самом начале и в самом конце и содержат в себе служебные данные для отображения стилей (в head содержится HTML файл с тегами <style>, <head>, <body> и подобные, в end – все закрывающие тэги).

После чего мы во временную переменную temp в цикле суммируем сначала нашую заголовочную переменную head, далее все вопросы и в конце суммируем переменную end. После этого мы для правильного вывода в браузер заменяем символы апострофов, а также пути директорий, для верного отображения изображений. Все изображения будут храниться в папке pic. И выводим содержимое переменной temp в браузер.

После заполнения формы тестированию, для завершения пользователю необходимо нажать кнопку «Завершить тестирование». Рассмотрим функцию этой кнопки.

Функция button5_Click – функция кнопки завершения тестирования. Проверяет ответы пользователя и выводит результат. В случае предварительного тестирования открывает сгенерированный учебник, в случае финального тестирования подсвечивает неверные ответы:

```
HtmlElementCollection inputs;
            inputs = webBrowser1.Document.GetElementsByTagName("input");
            double score = 0;
            switch (CurrentMode)
            {
                case "start_quest":
                    SQ_user_answers = new List<string>();
                    current_keys = new List<string>();
                    for (int i=0; i < SQ_count; i++)
                    {
                        SQ_user_answers.Add("");
                        current_keys.Add("");
                    }
                    for (int i = 0, k = 0; i < SQ_count; i++, k++)
                    {
                        if (inputs[k].GetAttribute("type") == "text")
SQ_user_answers[i]=inputs[k].GetAttribute("value");
                        else if (inputs[k].GetAttribute("type") == "radio")
                        {
                            for (int j = k; j < k + 4; j++)
                            {
                                if (inputs[j].GetAttribute("Checked") == "True")
                                {
                                   SQ_user_answers[i]=inputs[j].GetAttribute("value");
                                }
                            }
                            k = k + 3;
                        }
                    }
                    for (int i = 0; i < SQ_count; i++)
                    {
                        if (SQ_user_answers[i] == SQ_answers[i])
                        {
                            score += SQ_points[i];
                            current_keys.Add("-1");
                        }
                        else current_keys.Add(SQ_keys[i]);
                    }
                    SQ_TotalScore = SQ_sumpoints / 100;
                    SQ_TotalScore = Math.Floor(score / SQ_TotalScore);
```

```
                    label5.Text = SQ_TotalScore.ToString() + "/100";
                    MessageBox.Show("Поздравляем! Тест окончен. Количество набранных
баллов:\n" + SQ_TotalScore +
                        "/100\nСейчас будет автоматически сформированна и отрыта глава."
+
                        "\nНажмите ОК чтобы перейти к обучению.", "Тест окончен!",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    OpenParagraphs(false,true); // формирование и переход к учебнику
                    break;
                case "final_quest":

                    .............

                    break;
                default:
                    break;
            }
```

Как видно, функция достаточно объемная, поэтому здесь приведена только часть. Это связано с тем, что здесь будет происходить проверка всех ответов студента и подсчет баллов (как предварительного, так и итогового тестирования).

Первым делом мы загружаем все данные из окна браузера в переменную типа HtmlElementCollection (из неё мы извлечём ответы студента).

Далее с помощью конструкции switch – case мы проверяем, в каком режиме мы сейчас находимся (предварительного или итогового тестирования). Так как мы рассматриваем сейчас переход из предварительного тестирования, режим итогового рассматривать пока не будем.

Итак, если переменная текущего режима (CurrentMode) имеет значение "start_quest", то мы создаем переменные типа List для загрузки в них ответов студента и далее считываем в них ответы.

Мы проверяем аттрибуты переменной inputs, объявленной нами ранее, если находим аттрибут type = text, то значит, что мы нашли ответ пользователя на открытый вопрос, поэтому далее мы просто считаем то, что ввёл пользователь в наши переменные с ответом.

Если же мы находим аттрибут типа radio, значит, что это вопросы закрытого типа и тогда в переменную с ответом мы просто считываем номер того переключателя radio, которую пользователь отметил, как ответ.

После того, как мы считали ответы пользователя, мы сравниваем их с верными ответами, загруженными из базы данных. Если ответ верный, то в переменную с баллами добавляется количество баллов, соответствующее данному вопросу, а также значение ключа, соответствующее данному вопросу,

назначается значение «-1», что значит, что связанная глава пользователю показана не будет.

Если же пользователь ответил неверно, то баллы не начисляются, а в массив ключей добавляется ключ, соответствующий данному вопросу и пользователю будет отображена данная глава (если она есть).

Далее мы переводим баллы в 100-балльную систему и отображаем их пользователю в диалоговом окне. Как только пользователь закроет это окно, сразу же выполнится функция OpenParagraphs, с двумя параметрами false и true (о них далее).

Рассмотрим функцию загрузки и открытия параграфов OpenParagraphs.

Функция OpenParagraphs. Принимает в качестве параметров две переменных типа bool – full и generate. В зависимости от параметров, может открывать полную версию учебника, генерировать учебник на основе пройденного теста, а также открывать сохраненную версию учебника, если пользователь уже прошел предварительное тестирование и учебник был сгенерирован:

```
            if (!generate) {
                webBrowser1.DocumentText = SaveParagraphs;
                SetView("paragraph");
            }
            else {
                SQLiteCommand cmd = new SQLiteCommand(BD.conn);
                SQLiteDataReader r;
                cmd.CommandText = "SELECT * FROM [paragraphs] WHERE id =" +
IDs[comboBox1.SelectedIndex] + " ORDER BY CASE WHEN [index] IS NULL THEN 1 ELSE 0 END,
[index]";
                try
                {
                    P_text = new List<string>();
                    P_keys = new List<string>();
                    P_points = new List<double>();
                    r = cmd.ExecuteReader();
                    int j = 0;
                    foreach (DbDataRecord record in r)
                    {
                        P_text.Add("");
                        P_keys.Add("");
                        P_points.Add(0);

                        P_text[j] = r["text"].ToString();
                        P_keys[j] = r["key"].ToString();
                        P_points[j] = Convert.ToDouble(r["points"]);
                        j++;
                    }  }
                catch (SQLiteException ex) {
                    MessageBox.Show("Неверный файл учебника!\n\nОшибка:\n" + ex.Message,
"Ошибка открытия учебника", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                cmd.Dispose();
```

```
                // загрузка стилей
                SQLiteCommand cmd_styles = new SQLiteCommand(BD.conn);
                SQLiteDataReader r_styles;
                String P_head = "", P_end = "";
                cmd_styles.CommandText = "SELECT * FROM [paragraphs] WHERE
id='style_head' OR id='style_end'";
                try{
                    r_styles = cmd_styles.ExecuteReader();
                    foreach (DbDataRecord record in r_styles)
                    {
 if (r_styles["name"].ToString() == "head") P_head += r_styles["text"].ToString();
 if (r_styles["name"].ToString() == "end") P_end += r_styles["text"].ToString();
                    }
                }
                catch (SQLiteException ex)
                {
                    MessageBox.Show("Неверный файл учебника!\n\nОшибка:\n" + ex.Message,
"Ошибка открытия учебника", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                cmd_styles.Dispose();

                SetView("paragraph");

                if (full) {
                    String temp = P_head;
                    foreach (string s in P_text) temp += s;
                    temp += P_end;
                    temp = temp.Replace("'", "\"");
                    temp = temp.Replace("src=\"", "src=\"" + CurrentPath + "\\pic\\");
                    webBrowser1.DocumentText = temp;
                }
                else
                {
                    int n = P_text.Count();
                    String temp = P_head;
                    for (int i = 0; i < n; i++)
                    {
                        if (P_points[i] >= SQ_TotalScore) temp += P_text[i];
                        else
                        {
                            for (int j = 0; j < current_keys.Count(); j++)
                            {
                                if (P_keys[i] == current_keys[j])
                                {
                                    temp += P_text[i];
                                    break;
                                }
                            }
                        }
                    }
                    temp += P_end;
                    temp = temp.Replace("'", "\"");
                    temp = temp.Replace("src=\"", "src=\"" + CurrentPath + "\\pic\\");
                    webBrowser1.DocumentText = temp;
                    SaveParagraphs = temp;
                    label6.Text = "Да";
                    TestPass = true;
                }
            }
```

В эту функцию должно передаваться два параметра типа bool: full и generate. Дело в том, что пользователь может загрузить учебник не только по

итогу предварительного тестирования, но также из меню Настройки – Открыть полную версию учебника, где ему будут просто выданы все главы. Также пользователь может в главном окне, при загрузке учебника, не проходя предварительное тестирование, нажать на кнопку «Перейти к главе», в этом случае, ему также откроется полная версия.

Если же студент прошел предварительное тестирования, ему был сгенерирован учебник, он его прочитал и пошел проходить итоговое тестирование – он всегда может вернуться к сгенерированному для него учебнику по кнопке «Перейти к главе». В этом случае откроется не полная версия, а сохраненная.

Сначала мы проверяем значение переменной generate. Если она равна false (подразумевается, что генерация и загрузка учебника не нужна – пользователю уже был сгенерирован учебник), то в браузер загрузится переменная SaveParagraphs, в которой и хранится сохраненные главы и вид программы установится в режим “paragraphs”. Если generate = true, начинается загрузка и генерация учебника.

Для этого первым делом подключается база данных и из неё считываются все главы учебника в предварительно объявленные переменные. После чего аналогично происходит загрузка стилей (head и end, как и в случае предварительного тестирования).

Далее проверяется значение переменной full, которая передается функции также в качестве параметра. Если full = true, то мы не генерируем учебник для пользователя, а открываем полную версию, т.е. просто отображаем в браузер все главы подряд. Если full = false, то в переменную temp мы сначала записываем те, главы, у которых значение points меньше набранного значения пользователя, а после этого происходит загрузка тех глав, которые были связанны с вопросами предварительного тестирования ключами keys. В итоге пользователю будет отображены только те главы, которые соответствую набранным пользователем баллам и тем вопросам, на которые он не ответил.

Переменной TestPass присваиваем значение true, означающее, что тест пройден.

Рассмотрим функцию итогового тестирования (выполняется при нажатии кнопки «Перейти к итоговому тестированию»).

Функция OpenFQ – загружает и отображает вопросы итогового тестирования:

```
            SetView("final_quest");

            SQLiteCommand cmd = new SQLiteCommand(BD.conn);
            SQLiteDataReader r;
            cmd.CommandText = "SELECT * FROM [final_quest] WHERE id=" +
IDs[comboBox1.SelectedIndex] + " ORDER BY CASE WHEN [index] IS NULL THEN 1 ELSE 0 END,
[index]";
            try      {
                r = cmd.ExecuteReader();
                FQ_sumpoints = 0;
                FQ_questions = new List<string>();
                FQ_answers = new List<string>();
                FQ_points = new List<double>();
                int j = 0;
                foreach (DbDataRecord record in r)   {
                    FQ_questions.Add("");
                    FQ_answers.Add("");
                    FQ_points.Add(0);
                    FQ_questions[j] = r["text"].ToString();
                    FQ_answers[j] = r["answer"].ToString();
                    FQ_points[j] = Convert.ToDouble(r["points"]);
                    FQ_sumpoints += Convert.ToDouble(r["points"]);
                    j++;
                }
                FQ_count = j;
            }
            catch (SQLiteException ex)      {
                MessageBox.Show("Неверный файл учебника!\n\nОшибка:\n" + ex.Message,
"Ошибка открытия учебника", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            cmd.Dispose();

            // загрузка стилей
            SQLiteCommand cmd_styles = new SQLiteCommand(BD.conn);
            SQLiteDataReader r_styles;
            String FQ_head = "", FQ_end = "";
            cmd_styles.CommandText = "SELECT * FROM [final_quest] WHERE id='style_head'
OR id='style_end'";
            try            {
                r_styles = cmd_styles.ExecuteReader();
                foreach (DbDataRecord record in r_styles)
                {
  if (r_styles["name"].ToString() == "head") FQ_head += r_styles["text"].ToString();
  if (r_styles["name"].ToString() == "end") FQ_end += r_styles["text"].ToString();
                }
            }
            catch (SQLiteException ex)           {
                MessageBox.Show("Неверный файл учебника!\n\nОшибка:\n" + ex.Message,
"Ошибка открытия учебника", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            cmd_styles.Dispose();
            String temp = FQ_head;
            foreach (string s in FQ_questions) temp += s;
            temp += FQ_end;
            temp = temp.Replace("'", "\"");
            temp = temp.Replace("src=\"", "src=\"" + CurrentPath + "\\pic\\");
            webBrowser1.DocumentText = temp;
```

Сначала мы устанавливаем вид главного окна в режим “final_quest”.

Далее аналогично подключаем базу данных и выгружаем все данные о вопросах финального тестирования и стилях в предварительно объявленные переменные, после чего отображаем их пользователю в браузере.

Теперь рассмотрим ту часть кода функции кнопки «Завершить тестирование», которая относится к вопросам итогового тестирования.

Оставшаяся часть кода функции кнопки button5_Click. В случае итогового тестирования, проверяет ответы пользователя, выводит результат и подсвечивает неверные ответы:

```
            switch (CurrentMode)  {
                case "start_quest":

             ....................

                case "final_quest":
                    FQ_user_answers = new List<string>();
                    for (int i = 0; i < FQ_count; i++) FQ_user_answers.Add("");
                    for (int i = 0, k = 0; i < FQ_count; i++, k++)
                    {
                        if (inputs[k].GetAttribute("type") == "text") FQ_user_answers[i]
= inputs[k].GetAttribute("value");
                        else if (inputs[k].GetAttribute("type") == "radio")
                        {
                            for (int j = k; j < k + 4; j++)
                            {
                                if (inputs[j].GetAttribute("Checked") == "True")
                                {
                                    FQ_user_answers[i] = inputs[j].GetAttribute("value");
                                }
                            }
                            k = k + 3;
                        }
                    }
                    List<int> rightanswer = new List<int>();
                    for (int i = 0; i < FQ_count; i++)
                    {
                        if (FQ_user_answers[i] == FQ_answers[i])
                        {
                            score += FQ_points[i];
                            rightanswer.Add(1);
                        }
                        else rightanswer.Add(0);
                    }
                    FQ_TotalScore = FQ_sumpoints / 100;
                    FQ_TotalScore = Math.Floor(score / FQ_TotalScore);
                    label7.Text = FQ_TotalScore.ToString() + "/100";
MessageBox.Show("Поздравляем! Тест окончен. Количество набранных баллов:\n" +
FQ_TotalScore + "/100\nНеверные ответы будут подсвечены красным, нажмите ОК и внимательно
их изучите. " +
 "Если вы неудовлетворены своим результатом, прочитайте главу еще раз и повторите
тестирование.", "Тест окончен!", MessageBoxButtons.OK, MessageBoxIcon.Information);

                    for (int i = 0, k = 0; i < FQ_count; i++, k++)
                    {
                        if (inputs[k].GetAttribute("type") == "text")
                        {
                if (rightanswer[i] == 0) inputs[k].SetAttribute("className", "shadow");
```

```
                        else inputs[k].SetAttribute("className", "");
                    }
                    else if (inputs[k].GetAttribute("type") == "radio")
                    {
                        for (int j = k; j < k + 4; j++)
                        {
                            if (inputs[j].GetAttribute("Checked") == "True")
                            {
                                if (rightanswer[i] == 0)
inputs[j].SetAttribute("className", "shadow");
                                else inputs[j].SetAttribute("className", "");
                            }
                            else inputs[j].SetAttribute("className", "");
                        }
                        k = k + 3;
                    }
                }
                    break;
```

Здесь мы аналогично ситуации с вопросами предварительного тестирования в переменную inputs считываем ответы пользователя, считаем набранные баллы (в этот раз ключи мы не учитываем – в данном случае они без надобности) и отображаем их пользователю в диалоговом окне. Только после закрытия диалогового окна, в отличии от предварительного тестирования, мы в цикле пробегаем все значения аттрибутов типа “text” и “radio”, сравниваем с верным ответом еще раз и если ответ оказывается неверным, мы устанавливаем для этого элемента аттрибут “className” равный “shadow”. Все элементы с таким значением аттрибута будут подсвечены красным цветом. В данном случае, все неверные ответы пользователя подсветятся красным цветом.

На этом код, который относится к тестированию и отображению глав учебника окончен.

2.3.2.2 ФОРМА ОКНА ЗАГРУЗКИ УЧЕБНИКОВ С ОНЛАЙН-СЕРВЕРА (LoadOnlineForm.cs)

Эта форма может быть открыта из основного окна программы (Файл – Загрузить учебник онлайн...).

Все данные о серверах хранятся в файле servers.xml в папке с программой. Для удобства использования данные в этом файле хранятся в разметке XML.

Функция, выполняющаяся при загрузке формы.

Функция LoadOnlineForm_Load – загружает и отображает окно загрузки учебников с онлайн-сервера.

```
            if (config.AppSettings.Settings["SaveBookDir"] != null)
            {
                SaveBookDir = config.AppSettings.Settings["SaveBookDir"].Value;
            }
            if (config.AppSettings.Settings["WorkDir"] != null)
            {
                WorkDir = config.AppSettings.Settings["WorkDir"].Value;
            }

            xDoc = new XmlDocument();
            try
            {
                xDoc.Load("servers.xml");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Список серверов не загружен!\nПроверьте наличие файла
servers.xml\nОшибка:\n" + ex.Message, "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                Close();
                return;
            }
            LoadServersList();
```

Здесь первым делом из настроек извлекаются значения SaveBookDir и WorkDir, содержащие в себе пути сохранения учебника и временной директории соответственно.

Далее мы объявляем переменную xDoc типа XmlDocument и загружаем в неё файл со списком серверов servers.xml. После чего выполняем функцию, загружающую список серверов LoadServersList. Её и рассмотрим.

Функция LoadServersList – загружает список серверов:

```
            comboBox1.Items.Clear();
            XmlElement xRoot = xDoc.DocumentElement;
            XmlNodeList childnodes = xRoot.SelectNodes("//server/uri");
            foreach (XmlNode n in childnodes) comboBox1.Items.Add(n.InnerText);
            if (comboBox1.Items.Count!=0) comboBox1.SelectedIndex = 0;
```

В данной функции мы очищаем элемент comboBox1, объявляем переменную xRoot и присваиваем ей значение xDoc.DocumentElement (по сути это – элементы файла servers.xml, т.е. всё его содержимое). Далее объявляем переменную childnodes и присваиваем ей значение xRoot.SelectNodes("//server/uri"). Функция SelectNodes выбирает из наших элементов только те, которые имеют тег uri в родительском элементе server. В них содержатся адреса серверов.

Далее в цикле загружаем в comboBox1 все названия наших серверов и если есть хоть один, выбираем первый.

Тут же рассмотрим функции добавления и удаления серверов.

Функция button1_Click – функция кнопки «Добавить». Добавляет в список серверов новый адрес сервера:

```
            XmlElement serverElem = xDoc.CreateElement("server");
            XmlElement uriElem = xDoc.CreateElement("uri");
            XmlText uriText = xDoc.CreateTextNode(comboBox1.Text);
            uriElem.AppendChild(uriText);
            serverElem.AppendChild(uriElem);
            xRoot.AppendChild(serverElem);
            try  xDoc.Save("servers.xml");
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка сохранения:\n" + ex.Message, "Ошибка!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                Close();
                return;
            }
            comboBox1.Items.Clear();
            XmlNodeList childnodes = xRoot.SelectNodes("//server/uri");
            foreach (XmlNode n in childnodes)
            {
                comboBox1.Items.Add(n.InnerText);
                  }
```

Эта функция создает элементы типа XmlElements равные значениям server и uri, в uri мы добавляем значение comboBox1.Text, в которое пользователь ввел адрес своего сервера. Далее мы загружаем адрес сервера в переменную с uriElement, её в свою очередь в переменную serverElement, а ей в переменную xRoot, которая и представляет собой содержимое всего файла.

После этого сохраняем файл и перезагружаем все элементы серверов и отображаем их в comboBox1.

Функция button5_Click – функция кнопки удаления адреса сервера:

```
            foreach (XmlNode xnode in xRoot)
             {
                 foreach (XmlNode childnode in xnode.ChildNodes)
                 {
                     if (childnode.InnerText == comboBox1.Text)
                     {
                         xnode.RemoveAll();
                         xRoot.RemoveChild(xnode);
                         try
                         {
                             xDoc.Save("servers.xml");
                         }
                         catch (Exception ex)
                         {
                             MessageBox.Show("Ошибка
сохранения:\n" + ex.Message, "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                             Close();
                             return;
                         }
                         LoadServersList();
                         return;
                     }
                    }
```

В этой функции мы в цикле находим все адреса, значения которого равны comboBox1.Text, после чего удаляем их, сохраняем файл и перезагружаем список функцией LoadServersList.

Следующая функция выполняется при нажатии кнопки «Загрузить список». Она загружает список учебников, имеющихся на сервере.

Функция button4_Click – функция кнопки загруки списка учебников:

```
             Uri download;
             try
             {
                 download = new Uri(comboBox1.Text + "stb_list.xml");
             }
             catch (Exception ex)
             {
                 MessageBox.Show("Неверный формат адреса!\nАдрес должен быть вида:
http://site.com/", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                 return;
             }
             if (!Directory.Exists(WorkDir + "\\SmartTextBook"))
Directory.CreateDirectory(WorkDir + "\\SmartTextBook");
             WebClient client_serverlist = new WebClient();
             client_serverlist.DownloadProgressChanged += new
DownloadProgressChangedEventHandler(client_serverlist_DownloadProgressChanged);
             client_serverlist.DownloadFileCompleted += new
AsyncCompletedEventHandler(client_serverlist_DownloadFileCompleted);
```

```
            try
            {
                client_serverlist.DownloadFileAsync(download, WorkDir +
"\\SmartTextBook\\stb_list.xml");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка: " + ex.Message, "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
            Thread.Sleep(750);
                  LoadBooksList();
```

Тут мы объявляем переменную типа Uri, в которой будет храниться ссылка на файл, который хранится на сервере (на сервере должен находиться файл типа xml с названием stb_list.xml, хранящий список учебников).

Далее мы загружаем этот файл во временную папку и выполняем функцию загруки списка учебников LoadBooksList.

Функция LoadBooksList – загружает список книг с серверов:

```
            listBox1.Items.Clear();
            XmlDocument xDocList = new XmlDocument();
            try xDocList.Load(WorkDir + "\\SmartTextBook\\stb_list.xml");
            catch (Exception ex)          {
                MessageBox.Show("Ошибка! Не удается получить доступ к учебнику.\n\n" +
ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                Close();
                return;
            }
            XmlElement xRoot = xDocList.DocumentElement;
            XmlNodeList filenames = xRoot.SelectNodes("//book/filename");
            XmlNodeList descriptions = xRoot.SelectNodes("//book/description");
            FileList = new List<string>();
            List<string> DescriptionList = new List<string>();
            foreach (XmlNode n in filenames)
            {
                FileList.Add(n.InnerText);
            }
            foreach (XmlNode n in descriptions)
            {
                DescriptionList.Add(n.InnerText);
            }
            if (FileList.Count != DescriptionList.Count)
            {
                MessageBox.Show("Ошибка! Файл списка поврежден. Обратитесь к
администратору сервера.\n\n", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            int i = 0;
            foreach (String n in FileList)
            {
                listBox1.Items.Add(FileList[i] + " - " + DescriptionList[i]);
                i++;
            }
            if (listBox1.Items.Count != 0)
            {
                listBox1.SelectedIndex = 0;
```

```
            CurrentServer = comboBox1.Text;
            button3.Enabled = true;            }
```

Аналогично, как и в функции LoadServersList мы загружаем скачанный нами файл stb_list и извлекаем из него названия учебников и их описание в переменные filenames и descriptions типа XmlNodeList. Далее в цикле выводим в список listBox1 сначала имя файла и через тире его описание.

Если имеется хоть один элемент в списке, выделяем первый и активируем кнопку загрузки учебника.

Теперь посмотрим, что будет по нажатию кнопки «Загрузка».

Функция button3_Click – функция кнопки загрузки учебника:

```
            CurrentFile = FileList[listBox1.SelectedIndex];
            Uri download = new Uri(CurrentServer + "books/" + CurrentFile);
            if (!Directory.Exists(SaveBookDir)) Directory.CreateDirectory(SaveBookDir);
                String save = SaveBookDir + "\\" + CurrentFile;
            WebClient client = new WebClient();
            client.DownloadProgressChanged += new
DownloadProgressChangedEventHandler(client_DownloadProgressChanged);
            client.DownloadFileCompleted += new
AsyncCompletedEventHandler(client_DownloadFileCompleted);
            try client.DownloadFileAsync(download, save);
            catch (Exception ex)            {
                MessageBox.Show("Ошибка: " + ex.Message, "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);            }
                button3.Enabled = false;
```

Сначала мы присваиваем заранее объявленной переменной CurrentFile значение текущего файла (на основе выбранного в списке listBox1). Далее мы объявляем новую ссылку, которой присваиваем значение адреса выбранного учебника (учебник должен храниться на сервере в папке /books/имя_учебника.stb).

После чего мы объявляем переменную client типа WebClient, необходимую для загрузки файлов с сервера. Далее, мы устанавливаем для этой переменной значения DownloadProgressChanged и DownloadFileCompleted. Эти переменные хранят в себе ссылки на функции, которые будут выполняться при изменении прогресса загрузки и при выполнении загрузки. В качестве функций мы указываем функции, которые будут изменять наш прогресс-бар и функцию, которая запустится при завершении загрузки учебника.

Далее выполняем загрузку учебника, указав адрес файла и место для сохранения, полученное из настроек. В конце активируем кнопку открытия учебника. Функция этой кнопки очень проста.

Функция button6_Click – функция кнопки открытия загруженного учебника:

```
STBForm main = Owner as STBForm;
main.LoadBook(SaveBookDir + "\\" + CurrentFile);
main.SetView("main");
Close();
```

Мы объявляем переменную типа STBForm и назначем её родителем нашу главную форму. Через эту переменную мы можем обращаться к public-членам главной формы. Далее мы вызываем функцию родительской формы LoadBook в которой качестве параметра передаем только что загруженный учебник и устанавливаем вид родительской формы в значение “main”. После чего закрываем форму.

На этом рассмотрение всех функций и окон программы нашего учебника завершены (форма параметров рассмотрена не будет ввиду её тривиальности).

2.3.3 ПРОГРАММА РЕДАКТИРОВАНИЯ И СОЗДАНИЯ УЧЕБНИКОВ (STB_editor.exe)

2.3.3.1 ФОРМА ОСНОВНОГО ОКНА (EditorForm.cs)

Программа редактирования и создания учебников является полностью автономной и не зависит от основного учебника. Запускать её можно как напрямую по файлу STB_editor.exe, так и из окна учебника (Файл – Открыть в редакторе...), при условии, что файл редактора лежим в одной папке с основным учебником.

Первым делом рассмотрим функцию, выполняющуюся при загрузке формы редактора.

Функция Form1_Load – функция загрузки окна формы редактора. Отображает окно с приветствием:

```
            SetView("main");
            if (config.AppSettings.Settings["WorkDir"] != null)
                    CurrentPath = config.AppSettings.Settings["WorkDir"].Value;

            if (BD.IsSet())LoadListBooks();
            else  {
webBrowser1.DocumentText = "<style> body {background-color:#CAEBF3;color:#555;font-
family:Arial;font-size:24px;}</style>" +
"<body><div width = \"60%\" align = \"center\"><h1>Добро пожаловать!</h1><br><strong>" +
"Вы запустили редактор адаптивного учебника SmartTextBook.<br>" +
"Вы можете создать новый учебник:<br>Файл - Создать новый учебник.<br>" +
"<br>Также можете отредактировать существующий учебник:<br>Файл - Открыть
учебник.</strong></div><br>" +
"<div width = \"60%\" align = \"center\"> Система создана в качестве бакалаврской работы
"
+ "студентом СФУ Института Математики и Фундаментальной Информатики, <br> Сизых Я.В.
Группа ИМ12-06Б.<br><br><hr>2016г.</div></body>";
             if (comboBox1.Items.Count != 0) comboBox1.SelectedIndex = 0;   }
```

В редакторе также имеется функция SetView, которая активирует элементы формы в зависимости от переданного ей значения. При загрузке мы устанавливаем вид в основной режим (“main”).

Также в редакторе имеется аналогичный файл с классом BDClass (BDClass.cs), дополненный функциями сохранения файла.

В данной функции мы также загружаем файл настроек, проверяем, загружен ли файл учебника. Если загружен, вызываем функцию LoadListBooks

(аналогичная функция была рассмотрена ранее). Если файл не загружен, то просто выводим приветственное сообщение.

Рассмотрим функцию создания нового учебника.

Функция CreateBook – создает новый файл учебника и сохраняет его в указанном месте:

```
if (BD.IsSet()) BD.UnSet();
BD.CreateZIP(filename);
LoadBook(filename);
CurrentFile = filename;
```

В ней выгружается загруженный учебник и вызывается метод CreateZIP, который создаёт новый файл учебника и далее этот файл загружается функцией LoadBook, которая уже была рассмотрена ранее. Сейчас рассмотрим, как реализован метод CreateZIP класса BDClass.

Функция CreateZIP – метод класса BDClass для редактора, который создает новый архив из временной папки учебника и сохраняет его в указанном месте. В качестве параметра принимает текстовую строку с адресом файла:

```
zipPath = filepath;
if (File.Exists(zipPath)) File.Delete(zipPath);
if (Directory.Exists(extractPath))        {
    try Directory.Delete(extractPath, true);
    catch { }        }
Directory.CreateDirectory(extractPath);
Directory.CreateDirectory(extractPath + "\\pic");
String file = extractPath + "\\base.db";
SQL = "Data Source=" + file + "; Version=3;Journal Mode=Off";
conn = new SQLiteConnection(SQL);
try conn.Open();
catch (SQLiteException ex)     {
    MessageBox.Show(ex.Message, "Ошибка открытия учебника",
    MessageBoxButtons.OK, MessageBoxIcon.Error);         }

SQLiteCommand TableCreate = new SQLiteCommand(conn);
TableCreate.CommandText = "CREATE TABLE [books_list] (`id` INTEGER NOT NULL
PRIMARY KEY AUTOINCREMENT UNIQUE, `name` TEXT NOT NULL, `description` TEXT)";
TableCreate.ExecuteNonQuery();

TableCreate.CommandText = "CREATE TABLE [final_quest] (`text` TEXT NOT NULL,
`points` INTEGER NOT NULL, `answer` TEXT NOT NULL, `name` TEXT NOT NULL UNIQUE, `id`
TEXT, `index` INTEGER)";
TableCreate.ExecuteNonQuery();

TableCreate.CommandText = "CREATE TABLE [paragraphs] (`text` TEXT NOT NULL,
`points` INTEGER NOT NULL ,`key` TEXT NOT NULL, `name` TEXT NOT NULL UNIQUE, `id` TEXT,
`index` INTEGER)";
TableCreate.ExecuteNonQuery();

TableCreate.CommandText = "CREATE TABLE [start_quest] (`text` TEXT NOT NULL,
`points` INTEGER NOT NULL, `answer` TEXT NOT NULL, `key` TEXT NOT NULL, `name` TEXT NOT
NULL UNIQUE, `id` TEXT, `index` INTEGER)";
```

```
            TableCreate.ExecuteNonQuery();

            // добавляем стили
String HeaderText = "<style> table.table1{font-family: 'Trebuchet MS', sans-serif;font-
size: 16px;font-weight: bold;line-height: 1.4em;" +
"font-style: normal;border - collapse:separate;}.table1 tbody td{padding: 10px;text-
align:left;background-color:#DEF3CA;border: 2px solid #E7EFE0;" +
"-moz-border-radius:2px;-webkit-border-radius:2px;border-radius:2px;color:#666;text-
shadow:1px 1px 1px #fff;}</style>" +
"<table border width = 95% align = center class='table1'><tr><th>Вступительное
тестирование.</th></tr>";

            TableCreate.CommandText = "INSERT INTO [start_quest] (text, points, answer,
key, name, id, 'index') VALUES (@text, -1, -1, -1, \"head\", \"style_head\", -1)";
            TableCreate.Parameters.AddWithValue("@text", HeaderText);
            TableCreate.ExecuteNonQuery();

            String EndText = "</table>";
            TableCreate.CommandText = "INSERT INTO [start_quest] (text, points, answer,
key, name, id, 'index') VALUES (@text, -1, -1, -1, \"end\", \"style_end\", -1)";
            TableCreate.Parameters.AddWithValue("@text", EndText);
            TableCreate.ExecuteNonQuery();

            HeaderText = "<style> table.table1{font-family: 'Trebuchet MS', sans-
serif;font-size: 16px;font-weight: bold;line-height: 1.4em;" +
    "font-style: normal;border - collapse:separate;}.table1 tbody td{padding: 10px;text-
align:left;background-color:#CAD5F3;border: 2px solid #E7EFE0;" +
    "-moz-border-radius:2px;-webkit-border-radius:2px;border-radius:2px;color:#666;text-
shadow:1px 1px 1px #fff;}</style>" +
    "<table border width = 95% align = center class='table1'><tr><th>Итоговое
тестирование.</th></tr>";

            TableCreate.CommandText = "INSERT INTO [final_quest] (text, points, answer,
name, id, 'index') VALUES (@text, -1, -1, \"head\", \"style_head\", -1)";
            TableCreate.Parameters.AddWithValue("@text", HeaderText);
            TableCreate.ExecuteNonQuery();

            EndText = "</table>";
            TableCreate.CommandText = "INSERT INTO [final_quest] (text, points, answer,
name, id, 'index') VALUES (@text, -1, -1, \"end\", \"style_end\", -1)";
            TableCreate.Parameters.AddWithValue("@text", EndText);
            TableCreate.ExecuteNonQuery();

            HeaderText = "<body><div align=center>";

            TableCreate.CommandText = "INSERT INTO [paragraphs] (text, points, key, name,
id, 'index') VALUES (@text, -1, -1, \"head\", \"style_head\", -1)";
            TableCreate.Parameters.AddWithValue("@text", HeaderText);
            TableCreate.ExecuteNonQuery();

            EndText = "</table>";
            TableCreate.CommandText = "INSERT INTO [paragraphs] (text, points, key, name,
id, 'index') VALUES (@text, -1, -1, \"end\", \"style_end\", -1)";
            TableCreate.Parameters.AddWithValue("@text", EndText);
            TableCreate.ExecuteNonQuery();

            CreateBook("Новая глава");

            TableCreate.Dispose();

            conn.Close();
            SQLiteConnection.ClearAllPools();
            GC.Collect();
            GC.WaitForPendingFinalizers();
```

```
            if (File.Exists(zipPath)) File.Delete(zipPath);
            try ZipFile.CreateFromDirectory(extractPath, zipPath);
            catch (SQLiteException ex)  {
                MessageBox.Show(ex.Message, "Ошибка создания учебника:\n\n" + ex,
                MessageBoxButtons.OK, MessageBoxIcon.Error);            }
```

В этой функции мы первым делом создаём директорию во временной папке, в которой, в свою очередь, создаем директорию “pic” для хранения изображений.

Далее мы создаем SQL-запрос и подключаем базу данных. В результате этого в созданной нами папке создастся пока пустой файл базы данных “base.db”.

После этого мы последовательно создаем SQL-запросы для создания новых таблиц в нашей базе данных (таблицы books_list, start_quest, paragraphs и final_quest) с соответствующими значениями полей (которые обсуждались ранее в разделе с описанием структуры базы данных).

Далее мы заносим в каждую из таблиц по два элемента – head и end, в которые заносим текст с описанием стилей.

В конце мы освобождаем файл, закрываем базу данных, с помощью функции ZipFile.CreateFromDirectory создаем архив из нашей папки и сохраняем его в директории, которая была передана функции в качестве параметра. После этого удаляем временную папку и вызываем функцию CreateBook, которой передаем только что созданный файл.

Метод класса BDClass редактора – CreateBook. Создает новую главу учебника. В качестве параметра принимает текстовую строку и названием новой главы:

```
            SQLiteCommand BookCreate = new SQLiteCommand(conn);
            String WelcomeText = "<style> body {background-color:#CAEBF3;color:#555;font-
family:Arial;font-size:24px;}</style>" +
                                 "<body><div width = \"60%\" align = \"center\"><h1>Добро
пожаловать!</h1><br><strong>" +
                                 "Новая глава учебника.<br>" +
                                 "<br>Это заглавная страница новой
главы.</strong></div><br></body>";

            BookCreate.CommandText = "INSERT INTO [books_list] (name, description) VALUES
(@name, @string)";
            BookCreate.Parameters.AddWithValue("@string", WelcomeText);
            BookCreate.Parameters.AddWithValue("@name", BookName);
            BookCreate.ExecuteNonQuery();

                  BookCreate.Dispose();
```

Здесь мы создаем SQL-запрос, который создает новый элемент в таблице “books_list”, именует его name и создает стандартное описание. Название нового учебника передается функции в качестве параметра.

В итоге работы всех этих функций будет создан новый учебник.

Предположим, что пользователь открыл или создал новый учебник и он был загружен. Рассмотрим функции добавления новых глав (функции удаления и переименования аналогичные, рассмотрены не будут).

Функция button6_Click – функция кнопки добавления новой главы учебника:

```
            if (!BD.IsSet()) MessageBox.Show("Откройте учебник!", "Ошибка: учебник не
открыт", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            else
            {
                EnterBookName_Form dialog = new EnterBookName_Form();
                if (dialog.ShowDialog(this) == DialogResult.OK)
                {
                    String name = dialog.BookName;
                    BD.CreateBook(name);

                    LoadListBooks();
                    comboBox1.SelectedIndex = comboBox1.Items.Count - 1;
                    if (comboBox1.Items.Count != 0)
                    {
                        string text = Descriptions[comboBox1.SelectedIndex];
                        text = text.Replace("'", "\"");
                        text = text.Replace("src=\"", "src=\"" + CurrentPath +
"\\pic\\");
                        webBrowser1.DocumentText = text;
                    }

                    SetView("loaded");
                }
                dialog.Dispose();
                  }
```

Здесь пользователю выводится форма, в которую он должен ввести название новой главы. После этого вызывается рассмотренный нами метод CreateBook класса BDClass, в которую мы передаем введённое пользователем название. Далее перезагружаем список глав, выбираем последнюю главу в списке (она и будет той, которую мы только что добавили) и выводим описание этой главы в браузер. В конце переключаем режим окна в “loaded” (загружен).

Загрузив учебник и добавив новую главу, пользователь должен выбрать из списка категорий учебника, что он хочет добавить: элемент вступительного тестирования, параграфа или итогового тестирования. Как только пользователь

выбрал что-то из этого списка, будет вызвана функция LoadListElements, которая загрузит список элементов. Рассмотрим её.

Функция LoadListElements – функция загрузки списка элементов учебника:

```
        protected void LoadListElements()
        {
            listBox1.Items.Clear();

            switch(comboBox2.SelectedIndex)
            {
                case 0:
                    LoadSQ();
                    break;
                case 1:
                    LoadParagraph();
                    break;
                case 2:
                    LoadFQ();
                    break;
                default:
                    return;
            }
    if (listBox1.Items.Count > 0) listBox1.SelectedIndex = 0; }
```

В зависимости от того, какой элемент выбран, будет вызвана одна из функций: LoadSQ, LoadParagraph или LoadFQ. Все эти функции аналогичны таким из учебника и были уже расмотренны ранее.

Когда пользователь выбрал нужную категорию, он должен выбрать существующий элемент или добавить новый. При выборе существующего, программа загрузит данный элемент в браузер и все данные в соответствующие поля (название, ответ, баллы, ключ). Функция тривиальна и рассмотрена не будет. Рассмотрим функцию добавления нового элемента.

Функция кнопки button1_Click –добавляет новый элемент учебника:

```
        {
            if (!BD.IsSet()) MessageBox.Show("Откройте учебник!", "Ошибка: учебник не
открыт", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            else        {
                if (comboBox2.SelectedIndex==-1)
                {
MessageBox.Show("Выберите элемент учебника!", "Ошибка добавления", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    return;
                }
                EnterBookName_Form dialog = new EnterBookName_Form();
                if (dialog.ShowDialog(this) == DialogResult.OK)
                {
                    String name = dialog.BookName;
                    if (listBox1.FindStringExact(name) != -1)
                    {
MessageBox.Show("Вы должны указать уникальное имя для каждого элемента!", "Ошибка
```

```
добавления", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                        return;                    }
                    SQLiteCommand ElementAdd = new SQLiteCommand(BD.conn);
                    switch (comboBox2.SelectedIndex)
                    {
                        case 0:
                            ElementAdd.CommandText = "INSERT INTO [start_quest] (text,
points, answer, key, name, id, 'index') VALUES (\"\", -1, \"Ответ\", -1, @name, @id, 1)";
                            ElementAdd.Parameters.AddWithValue("@name", name);
                            ElementAdd.Parameters.AddWithValue("@id",
IDs[comboBox1.SelectedIndex]);
                            ElementAdd.ExecuteNonQuery();
                            LoadListElements();
                            listBox1.SelectedIndex = listBox1.Items.Count - 1;
                            break;
                        case 1:
                            ElementAdd.CommandText = "INSERT INTO [paragraphs] (text,
points, key, name, id, 'index') VALUES (\"\", -1, -1, @name, @id, 1)";
                            ElementAdd.Parameters.AddWithValue("@name", name);
                            ElementAdd.Parameters.AddWithValue("@id",
IDs[comboBox1.SelectedIndex]);
                            ElementAdd.ExecuteNonQuery();
                            LoadListElements();
                            listBox1.SelectedIndex = listBox1.Items.Count - 1;
                            break;
                        case 2:
                            ElementAdd.CommandText = "INSERT INTO [final_quest] (text,
points, answer, name, id, 'index') VALUES (\"\", -1, \"Ответ\", @name, @id, 1)";
                            ElementAdd.Parameters.AddWithValue("@name", name);
                            ElementAdd.Parameters.AddWithValue("@id",
IDs[comboBox1.SelectedIndex]);
                            ElementAdd.ExecuteNonQuery();
                            LoadListElements();
                            listBox1.SelectedIndex = listBox1.Items.Count - 1;
                            break;
                        default:
                            MessageBox.Show("Выберите элемент учебника!", "Ошибка
добавления", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                            return;
                    }
                    ElementAdd.Dispose();

                }
                dialog.Dispose();
            }
```

Здесь мы сначала предлагаем пользователю ввести название нового элемента, далее проверяем, какая категория у нас выбрана. В зависимости от выбранной категории с помощью SQL-запросов мы добавляем в соответствующие таблицы новые элементы с заданным названием.

В конце мы перезагружаем список элементов и выбираем последний из них.

Создав элемент, пользователь может изменить его название, ответ, баллы, ключ или порядковый номер. Сохранение введёных пользователем данных будет происходить после того, как он переместит фокус с поля ввода на любой другой элемент. Рассмотрим функцию сохранения введёных пользователем

данных на примере поля «Название элемента» (остальные поля сохраняются абсолютно аналогичным образом).

Функция textBox1_Leave – функция, выполняющаяся при покидании поля ввода. Сохраняет текст, введенный пользователем в базу данных:

```
                if (textBox1.Text == "")
                {
                    MessageBox.Show("Введите название!", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    return;
                }
                if (listBox1.FindStringExact(textBox1.Text) != listBox1.SelectedIndex)
                {
                    MessageBox.Show("Вы должны указать уникальное имя для каждого
элемента!", "Ошибка добавления", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }
                SQLiteCommand NameAdd = new SQLiteCommand(BD.conn);
                String name = textBox1.Text;
                String type = "";
                switch (comboBox2.SelectedIndex)
                {
                    case 0:
                        type = "start_quest";
                        break;
                    case 1:
                        type = "paragraphs";
                        break;
                    case 2:
                        type = "final_quest";
                        break;
                    default:
                        break;
                }
                NameAdd.CommandText = "UPDATE [" + type + "] SET `name` = @name WHERE
name = '" + listBox1.Text + "'";
                NameAdd.Parameters.AddWithValue("@name", name);
                NameAdd.ExecuteNonQuery();
                NameAdd.Dispose();
                int currentIndex = listBox1.SelectedIndex;
                LoadListElements();
                listBox1.SelectedIndex = currentIndex;
```

Первым делом происходит проверка, если пользователь оставил поле пустым. В случае, если пользователь ничего не ввёл, ему выводится предупреждение. Иначе формируется SQL-запрос, в котором мы обновляем элемент в выбранной таблице (start_quest, paragraphs или final_quest), сохраняя новое имя элемента. Далее мы перезагружаем список элементов, при этом запоминая текущую позицию (чтобы пользователю не приходилось каждый раз выбирать редактируемый элемент).

Также в редакторе предусмотрена возможность редактирования HTML-текста вопроса напрямую, хоть это и не рекомендуется. Рассмотрим эту функцию.

Функция кнопки button3_Click – отображает HTML-код:

```
            if (!BD.IsSet()) MessageBox.Show("Откройте учебник!", "Ошибка: учебник не
открыт", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            else            {
       MessageBox.Show("Внимание! Прежде чем редактировать HTML текст напрямую,
убедительная просьба ознакомиться со справкой!", "Внимание!", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                button3.Visible = false;
                webBrowser1.Visible = false;
                button11.Visible = true;
                richTextBox1.Visible = true;
                if (CurrentMode != "loaded")
                {
                    switch (CurrentMode)
                    {
                        case "start_quest":
                            richTextBox1.Text = SQ_questions[listBox1.SelectedIndex];
                            break;
                        case "paragraph":
                            richTextBox1.Text = P_text[listBox1.SelectedIndex];
                            break;
                        case "final_quest":
                            richTextBox1.Text = FQ_questions[listBox1.SelectedIndex];
                            break;
                        default:
                            break;
                    }
                }
                else if (CurrentMode == "loaded") richTextBox1.Text =
Descriptions[comboBox1.SelectedIndex];
            }
```

По нажатию на кнопку «Показать HTML-код», пользователю выведется сообщение о том, что редактировать код напрямую не рекомендуется. После этого элемент браузера скроется с формы и отобразится элемент RichTextBox, в который загрузятся вопросы или главы в зависимости от выбранного элемента в listBox1. Также отобразится кнопка «Возрат», которая вернет браузер и скроет поле ввода.

Сохранение HTML-кода происходит тогда, когда пользователь покидает поле ввода RichTextBox и просходит аналогично, как и в случае с вводом названия элемента, ответа, количества баллов или порядкового номера.

2.3.3.2 ФОРМА ОКНА РЕДАКТИРОВАНИЯ ЭЛЕМЕНТА (QuestionAdd.cs)

В это окно пользователь попадает при нажатии на кнопку «Редактировать вопрос» или «Редактировать текст» (в зависимости, что он редактирует – элемент вопроса или элемент главы). Мы рассмотрим случай, когда пользователь редактирует элемент вопроса. Редактирование текста происходит аналогичным образом.

Основной код этой формы выполняется при загрузке формы. Рассмотрим его.

Функция QuestionAdd_Load – загрузка формы окна редактирования нового вопроса:

```
            switch (main.CurrentMode)  {
                case "start_quest":
                    text = main.SQ_questions[main.listBox1.SelectedIndex];
                    break;
                case "final_quest":
                    text = main.FQ_questions[main.listBox1.SelectedIndex];
                    break;
                default:
                    break;
            }
            if (text.Contains("radio")) radioButton1.Checked = true;
            else radioButton2.Checked = true;
            string QuestText = "";
            string pattern = @"(?<=<td>)[\d\D]+(?=<form name)";
            RegexOptions option = RegexOptions.IgnoreCase;
            Regex newReg = new Regex(pattern, option);
            MatchCollection matches = newReg.Matches(text);

            if (matches.Count > 0 ) QuestText = matches[0].Value;

            QuestText = QuestText.Replace("\n","");
            QuestText = QuestText.Replace("<br>", "\n");
            richTextBox1.Text = QuestText;

            if (QuestType == "close")
            {
                pattern = @"(?<=(value=""\d"">|value='\d'>)).+(?=<br>)";
                newReg = new Regex(pattern, option);
                matches = newReg.Matches(text);
                if (matches.Count==4)
                {
                    textBox2.Text = matches[0].Groups[0].Value;
                    textBox3.Text = matches[1].Groups[0].Value;
                    textBox4.Text = matches[2].Groups[0].Value;
                    textBox5.Text = matches[3].Groups[0].Value;
                }
                else
                {
textBox2.Text = "Проблемы с распознаванием ответа. Впишите его сюда заново и
сохранитесь.";
textBox3.Text = "Проблемы с распознаванием ответа. Впишите его сюда заново и
сохранитесь.";
textBox4.Text = "Проблемы с распознаванием ответа. Впишите его сюда заново и
```

```
сохранитесь.";
textBox5.Text = "Проблемы с распознаванием ответа. Впишите его сюда заново и
сохранитесь.";
                }

.................
}
            if (QuestType == "open")
            {
                switch (main.CurrentMode)
                {
                    case "start_quest":
                        textBox6.Text = main.SQ_answers[main.listBox1.SelectedIndex];
                        break;
                    case "final_quest":
                        textBox6.Text = main.FQ_answers[main.listBox1.SelectedIndex];
                        break;
                    default:
                        break;        }
```

Первым делом в переменную text заносим текст вопроса. Если в тексте найдено ключевое слово «radio», значит мы имеем дело с вопросом закрытого типа и устанавливаем переключатель radioButton1.Checked = true (в этом случае отобразятся поля для ввода 4-х вариантов ответа, иначе будет показано поле ввода ответа для открытого текста – пользователь, меняя эти переключатели, может выбирать, какого типа вопрос он хочет сделать).

Далее нам необходимо произвести синтаксический анализ вопроса и выделить из него текст вопроса и, собственно, варианты ответа. Для этого используются так называемые регулярные выражения. Регулярные выражения позволяют просто и быстро произвести синтаксический анализ текста по заданному шаблону и найти все подстроки текста, соответствующие шаблону.

В нашем случае шаблон для поиска текста такой: (?<=<td>)[\d\D]+(?=<form name). В HTML-файле текст начинается от тэга <td> и шаблон ищет все символы, начиная от этого тэга ((?<=<td>) – начиная от тэга <td>, [\d\D]+ - найти весь текст). Текст в нашем коде будет продолжаться до тэга <form name=...>. Поэтому текст будем искать до этого тэга - (?=<form name).

Далее мы создаем новое регулярное выражение с заданной опцией игнорирония регистра и нашим шаблоном. Найденный по этому шаблону текст мы сохраняем в переменную QuestText.

Шаблон для поиска вариантов ответа следующий: (?<=(value=""\d"">|value='\d'>)).+(?=
). Здесь мы ищем все символы от тэга, оканчивающегося на ...value=”\d”> или на ...value=’\d’>, где \d – любая цифра, до тэга
. Все найденные вопросы сохраняются в соответствующие поля.

Если соответствий в тексте не найдено, то в поля будет выдано сообщения о проблеме распознавания. Это может быть следствием неверного редактирования HTML-кода пользователем напрямую.

Далее, если вопрос открытого типа, то текст ответа мы просто берем из родительской формы соответствующего массива с ответами и заносим в поле с ответом.

Также в этом окне пользователь может вставить изображение в текст, нажав на кнопку «Вставить изображение».

Функция кнопки button3_Click – загружает и вставляет в текст новое изображение:

```
OpenFileDialog dialog = new OpenFileDialog();
dialog.Filter = "Изображения|*.jpg; *.jpeg; *.png; *.bmp; *.gif";
if (dialog.ShowDialog() == DialogResult.OK)
{
    Form1 main = Owner as Form1;
    string filename = dialog.SafeFileName;
    while (File.Exists(main.BD.extractPath + "\\pic\\" + filename))
        filename = "1" + filename;

    File.Copy(dialog.FileName, main.BD.extractPath + "\\pic\\" + filename);
    richTextBox1.Text += "\n<img src=\"" + filename + "\">\n";       }
```

Здесь пользователю предлагается диалоговое окно с открытием файла, где он может выбрать любое изображение. После этого изображение будет сохранено в папке /pic. Если изображение с таким названием уже будет существовать в папке, то к началу имени будет добавлена единица. После этого в текст вопроса будет добавлена строка с тэгами для отображения изображения.

По нажатию кнопки OK программа сохранит все введёные данные в базе данных и закроет окно.

2.3.3.3 ФОРМА ОКНА РЕДАКТИРОВАНИЯ СТИЛЕЙ (StylesForm.cs)

Данное окно выводится пользователю, когда он выбирает пункт меню Настройки – Выбрать стиль оформления... .

Функция StylesForm_Load – загрузка формы окна редактора стилей:

```
            cmd_styles.CommandText = "SELECT * FROM [start_quest] WHERE id='style_head'
OR id='style_end'";
            try  {
            SQ_head = "";
            SQ_end = "";
            cmd_styles.CommandText = "SELECT * FROM [start_quest] WHERE id='style_head'
OR id='style_end'";
            try   {
                r_styles = cmd_styles.ExecuteReader();
                foreach (DbDataRecord record in r_styles)
                {
if (r_styles["name"].ToString() == "head") SQ_head += r_styles["text"].ToString();
if (r_styles["name"].ToString() == "end") SQ_end += r_styles["text"].ToString();  }
            }
            catch (SQLiteException ex)     {
                MessageBox.Show("Неверный файл учебника!\n\nОшибка:\n" + ex.Message,
"Ошибка открытия учебника", MessageBoxButtons.OK, MessageBoxIcon.Error);
                Close();      }
            cmd_styles.Dispose();

      ..... // аналогично загружаются стили параграфов и итогового тестирования

            text = "<tr><td>Пример текста</td></tr>";
```

В данном коде с помощью SQL-запросов мы загружаем стили всех категорий учебника (начального тестирования, итогового и параграфов).

Далее мы задаём значение переменной text, которая будет выведена в браузере в качестве примера текста.

Первым делом пользователю надо выбрать категорию, стиль отображения которой он хочет изменить. Рассмотрим код, выполняющийся при изменении текущей категории.

Функция comboBox1_SelectedIndex Changed – функция, выполняющаяся при изменении выбранного индекса выпадающего списка с категориями учебника:

```
            textBox1.Enabled = true;
            button4.Enabled = true;
            button3.Enabled = true;
            listBox1.Enabled = true;
            textBox1.Enabled = true;
            webBrowser1.Visible = true;
            richTextBox1.Visible = false;
            button4.Visible = true;
```

```
button5.Visible = false;
string pattern = @"(?<=<th>)[\d\D]+(?=</th>)";
RegexOptions option = RegexOptions.IgnoreCase;
Regex newReg = new Regex(pattern, option);
MatchCollection matches;

string temp = "";
switch (comboBox1.SelectedIndex)     {
    case 0:
        temp = SQ_head + text + SQ_end;
        break;
    case 1:
        temp = P_head + text + P_end;
        break;
    case 2:
        temp = FQ_head + text + FQ_end;
        break;
    default:
        break;
}
matches = newReg.Matches(temp);
if (matches.Count > 0) header = textBox1.Text = matches[0].Value;
else header = textBox1.Text = "";
webBrowser1.DocumentText = temp;
```

Первым делом делаем активными элементы формы. Затем с помощью регулярных выражений ищем заголовок между тэгами <th> и </th>. После чего выводим наш образец текста в браузер вместе со стилями и выводим найденный с помощью регулярных выражений заголовок.

В данном окне также присутствует возможность прямого редактирования HTML-кода, он аналогичен предыдущему и рассмотрен не будет. Равно, как и код изменения заголовка.

Помимо прочего, здесь же присутствует кнопка выбора фонового цвета. Вот этот код мы рассмотрим.

Функция кнопки button3_Click – открывает диалоговое окно выбора цвета и устанавливает новый фоновый цвет:

```
ColorDialog dialog = new ColorDialog();
dialog.ShowDialog();
string g = Convert.ToString(dialog.Color.G, 16);
string b = Convert.ToString(dialog.Color.B, 16);
string r = Convert.ToString(dialog.Color.R, 16);
string color = "#" + r + g + b;

string pattern = @"(?<=background-color:)[\d\D]+(?=;)";
RegexOptions option = RegexOptions.IgnoreCase;
Regex newReg = new Regex(pattern, option);
string temp = "";
switch (comboBox1.SelectedIndex)    {
    case 0:
        SQ_head = newReg.Replace(SQ_head, color);
        temp = SQ_head + text + SQ_end;
        break;
    case 1:
```

```
                SQ_head = newReg.Replace(P_head, color);
                temp = P_head + text + P_end;
                break;
            case 2:
                SQ_head = newReg.Replace(FQ_head, color);
                temp = FQ_head + text + FQ_end;
                break;
            default:
                break;    }
        webBrowser1.DocumentText = temp;
```

Здесь пользователю отобразится диалоговое окно выбора цвета. После того, как пользователь выберет желаемый цвет, мы конвертируем значения красного, синего и зеленого (RGB) в шестнадцатиричную систему (цвета в HTML хранятся именно в этой системе) и объявляем переменную color, в который суммируем значения каждого цвета.

Далее с помощью всё тех же регулярных выражений мы с помощью функции Replace заменяем в заголовке цвет по шаблону. Цвет ищем между строкой background-color: и знаком «;». После этого выводим текст в браузер.

По нажатию на кнопку OK программа сохранит измененные стили (аналогичный код уже был рассмотрен ранее).

На этом мы рассмотрели весь код учебника и редактора.

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для работы с системой адаптивного обучения первым делом необходимо запустить исполняемый файл программы “SmartTextBook.exe”.

Система имеет интуитивно-понятный интферфейс. При запуске отобразится главное окно с приветствием. Чтобы открыть учебник, нужно выбрать пункт меню «Файл – Открыть учебник...» и выбрать нужный файл.

Если вы хотите загрузить учебник с облака, необходимо выбрать пункт меню «Файл – Загрузить учебник онлайн...», в появившемся окне выбрать нужный вам сервер, нажать кнопку «Загрузить список». В списке появятся все учебники, хранимые на сервере. Из этого списка вам нужно выбрать необходимый вам учебник и нажать кнопку «Загрузка». Начнется загрузка (за процессом загрузки можно следить по прогресс-бару) и по её окончанию вы можете открыть загрузившийся учебник, нажав на кнопку «Открыть».

После открытия учебника появится основное окно программы с описанием учебника. Пользователь может выбрать главу в верхнем выпадающем списке и нажав кнопку «Открыть».

После выбора главы пользователю предлагается пройти вступительное тестирование, открыть главы учебника или пройти итоговое тестирование. Пользователь может выбрать любое из этих действий, но первым делом рекомендуется пройти предварительное тестирование, для чего, соответственно, нужно нажать кнопку «Перейти к вступительному тестированию».

Пользователь попадет на форму с тестом, который ему нужно пройти, отвечая на вопросы двух типов: либо тестовый вопрос с четырьмя вариантами ответа, либо вопрос со свободной формой ответа, где пользователю нужно отвечать, вводя ответ в текстовое поле.

После того, как пользователь ответил на предложенные вопросы, нужно нажать кнопку «Завершить тестирование», после чего пользователю сообщится количество набранных баллов и в форме окна отобразится учебник, который был сформирован на основе пройденного теста.

Если пользователь не желает проходить начальное тестирование, можно открыть полную версию учебника из основного окна программы, выбрав пункт меню «Настройки – Открыть полную версию учебника».

После прочтения материала, чтобы проверить знания, необходимо нажать кнопку «Перейти к итоговому тестированию», где пользователю будет предложен тест, подобный начальному тестированию, после прохождения которого пользователю будет сообщено количество набранных баллов, а также, в отличии от начального тестирования, неверные ответы будут подсвечены красным цветом. Пользователь может проанализировать свои неверные ответы и вернуться к учебнику (нажав кнопку «Перейти к главе»), чтобы более подробно изучить материал.

На этом изучение учебника заканчивается. Рассмотрим, как можно отредактировать или создать новый учебник.

Для редактирования или создания учебника необходимо либо выбрать пункт меню «Файл – Открыть в редакторе», в этом случае в редакторе откроется текущий учебник, если он открыт, либо открыть программу редактора отдельно, запустив исполняемый файл STB_editor.exe.

В левой части окна приводится список всех элементов учебника. Всего типов элемента может быть три: вопрос начального тестирования, вопрос итогового тестирования, а также глава учебника. Выбрать необходимую категорию можно из выпадающего списка слева. Внизу имеются две кнопки, позволяющие создавать или удалять элементы учебника.

Нужно отметить, что для правильно работы программы, очень важно, чтобы каждый элемент вопроса содержал только один вопрос одного типа.

В верхней части можно редактировать название элемента – это нужно для удобного ориентирования автора, в самому учебнике пользователю эти названия не отображаются.

Основную часть окна занимает окно браузера, отображающее содержание главы учебника или вопроса.

Для редактирования учебника первым делом необходимо открыть нужный вам учебника через пункт меню «Файл – Открыть учебник...» или же вы можете создать новый учебник, выбрав пункт меню «Файл – Создать новый учебник...». В диалоговом окне вам нужно будет выбрать файл учебника или указать место, куда сохранить новый.

После открытия учебника в редакторе, вам необходимо выбрать одну из существующих глав или добавить новую в верхнем выпадающем списке. Также вы можете удалить главу или переименовать её, используя кнопки справа от

списка. Выбрав главу, нужно нажать на кнопку «Открыть», после чего содержимое главы будет загружено.

Открыв учебник и выбрав нужную вам главу необходимо выбрать одну из трёх категорий учебника из выпадающего списка слева, затем выбрать необходимый элемент категории или добавить новый, нажав на кнопку «Добавить элемент». Удалить элемент можно нажав на кнопку «Удалить элемент».

Ответы на вопросы, также как количество баллов за верный ответ, а также ключ для связи главы и вопроса можно ввести в полях под формой. Там же можно изменить порядковый номер элемента. Ответ на тестовый вопрос нужно вводить цифрой – от одного до четых (1-4), в зависимости от того, какой из четырех варианта ответа верный. На текстовый вопрос ответ вводится в виде произвольной текстовой строки.

Количество баллов позволяется вводить в любой системе (пятибалльной, стобалльной, какой угодно) – программа в итоге отобразит их студенту в пересчете из 100.

Ключ связи нужен для того, чтобы при неправильном ответе на вопрос с таким идентификатором, глава учебника, в которой прописан такой же идентификатор, отобразилась пользователю в независимости от набранных баллов. Указывать необязательно. Если не желаете указывать ключ, оставьте значение равным «-1». Писать ключ желательно латинскими символами и цифрами без пробелов.

Вы можете редактировать текст вопроса или главы напрямую через HTML, нажав на кнопку «Показать HTML-код», но так делать не рекомендуется, о чем программа вас предупредит. Редактирование вопросов и текста главы лучше производить, нажав на кнопку «Редактировать вопрос» или «Редактировать текст» соответственно. Очистить текст можно нажав на кнопку «Очистить».

Нажав на кнопку «Редактировать вопрос» откроется окно редактирования вопроса. В окне вы можете отредактировать текст вопроса, выбрать его тип (открытый – свободная форма ответа и закрытый – выбор ответа из 4-х вариантов). Выбрав тип вопроса, заполните поля с вариантами ответов (или поле ответа, если вопрос открытого типа) и укажите верный ответ переключателем справа (это в случае вопроса закрытого типа).

Здесь же вы можете вставить в текст вопроса изображение. Для этого нажмите кнопку «Добавить изображение», в открывшемся окне выберете нужное изображение. В текст вопроса добавится тэг <img src=”...”> - в этом месте будет располагаться выбранное вами изображение. Для удаления изображения или перенос его в новое место – удалите или переместите этот тэг. Для завершения редактирования вопроса нажмите кнопку ОК.

Для редактирования текста главы нажмите кнопку «Редактировать текст». Откроется аналогичное окно, но без вариантов ответа. В нем можете отредактировать текст, добавить изображение. По окончанию редактирования также необходимо нажать кнопку ОК.

Ключ связи необязателен, но если пользователь не ответит на вопрос с таким же идентификатором – система выведет эту главу в независимости от набранных баллов. В графе количества баллов, в случае главы учебника, указывается суммарное количество баллов, которое должен набрать студент, чтобы эта глава НЕ отображалась студенту. Допустим, если студент набрал 70 и более баллов, то глава учебника с таким количеством баллов не будет ему показана. Если в главе учебника стоит 0 баллов, то эта глава будет показана только в том случае, если пользователь не ответил ни на один вопрос предварительного тестирования. Если же, например, указано максимально возможное количество баллов или больше, то эта глава будет показана студенту в независимости от того, сколько баллов он набрал на тестировании.

Вы также можете отредактировать стиль отображения вопросов или глав учебника. Для этого выберите пункт меню «Настройки – Выбрать стиль оформления...». Откроется окно, в котором вам надо выбрать из выпадающего списка, для какой категории учебника вы хотите изменить стиль. Выбрав категорию, в окне браузера отобразится пример оформления текста. Вы можете отредактировать заголовок, а также выбрать цвет фона, нажав на кнопку «Изменить цвет». Из списка слева вы можете выбрать стиль оформления из доступных в редакторе. Здесь же вы можете напрямую отредактировать стили через HTML, нажав кнопку «Редактор HTML». Для завершения редактирования нажмите кнопку ОК.

Для сохранения учебника необходимо выбрать пункт меню «Файл – Сохранить». Либо же, если пользователю необходимо сохранить учебник в новом месте, можно выбрать пункт меню «Файл – Сохранить как...» и выбрать место и имя для нового учебника. Из этого же пункта меню можно открыть созданный вами учебник в программе учебника, выбрав пункт «Открыть в SmartTextBook...». Ваш учебник сохранится и откроется в программе учебника.

4 РЕЗУЛЬТАТЫ

В результате дипломной работы была написана программа, в которой были реализованы все запланированные возможности. Также был создан редактор собственных учебников.

Ниже приведены скриншоты всех окон программы и редактора.

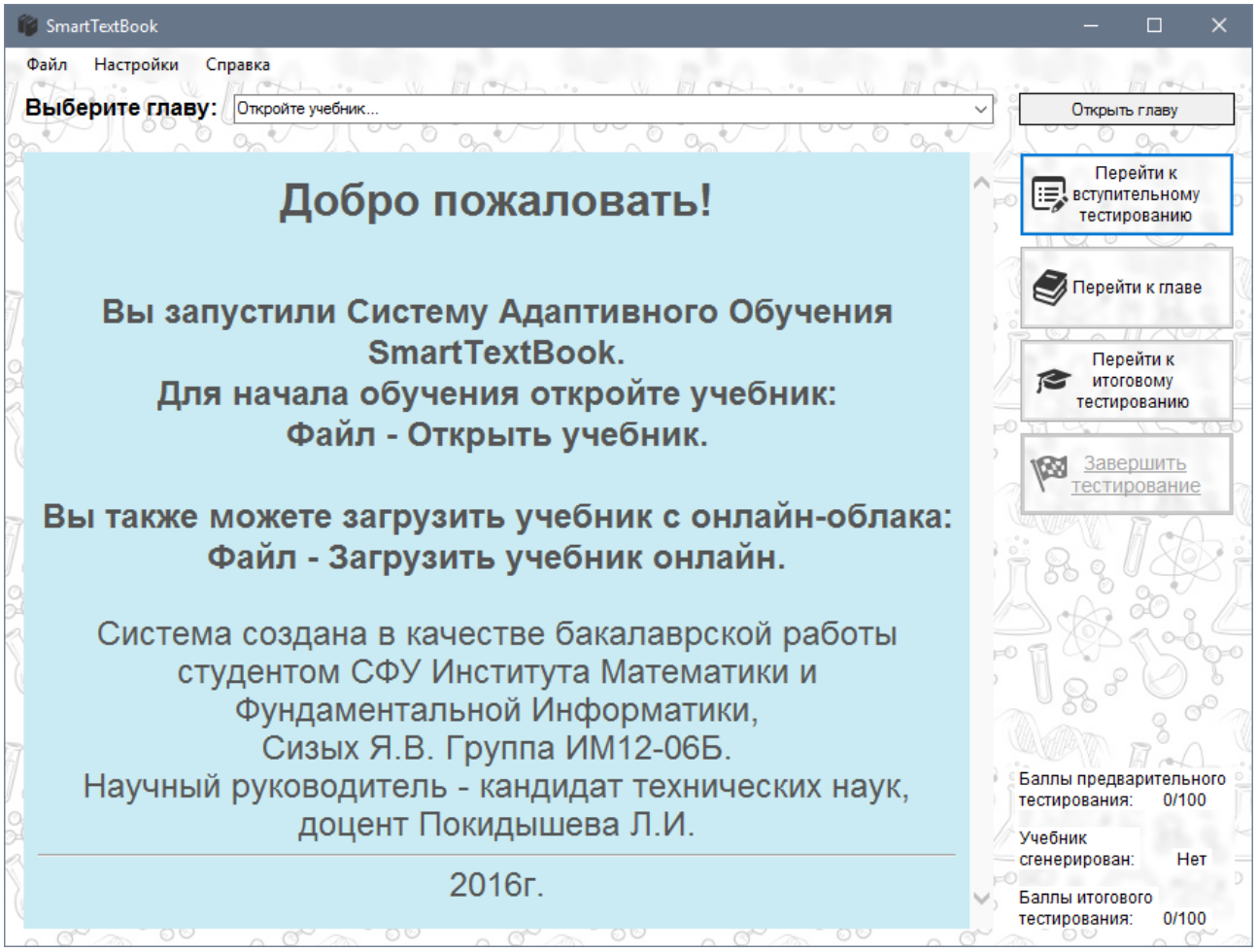

Рисунок 3.1 – Стартовое окно

В качестве примера была написан краткий курс по матрицам.

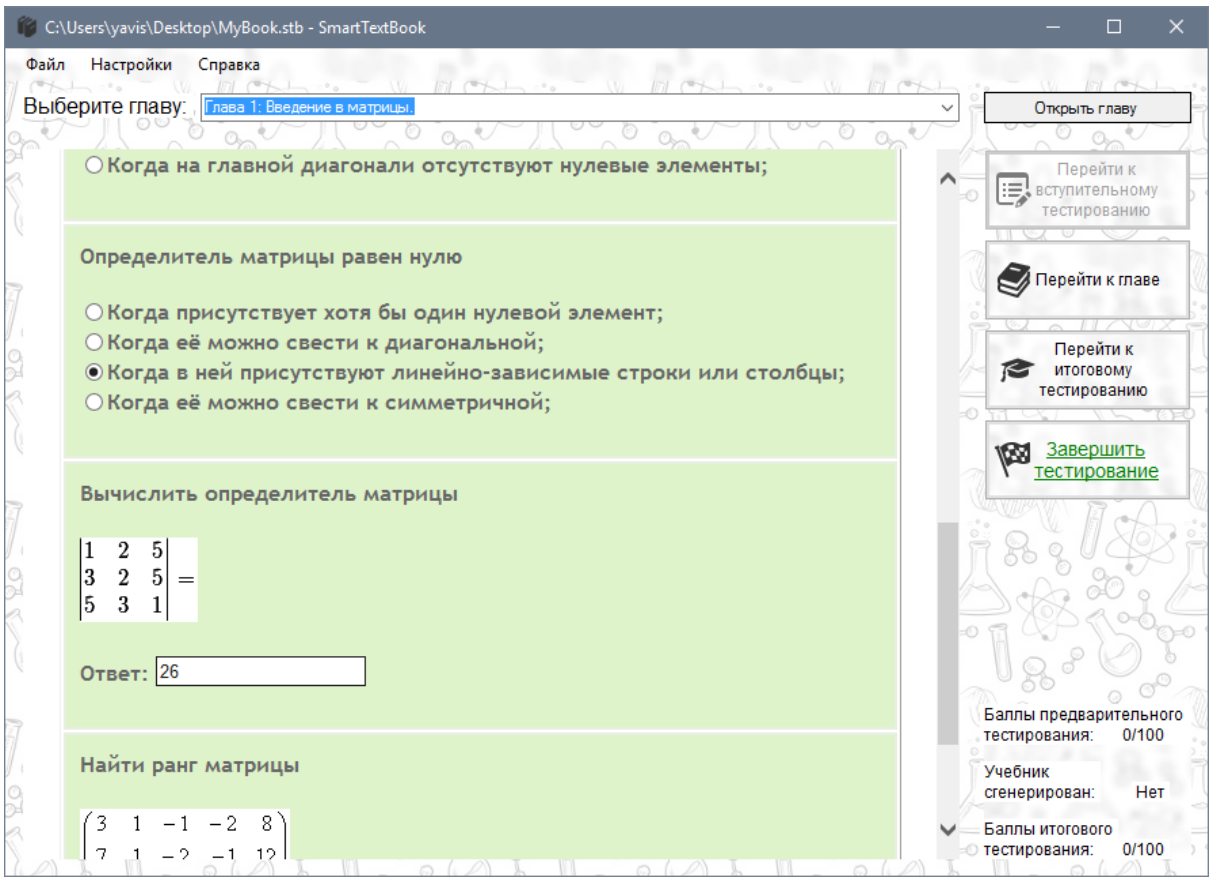

Рисунок 3.3 – Окно предварительного тестирования

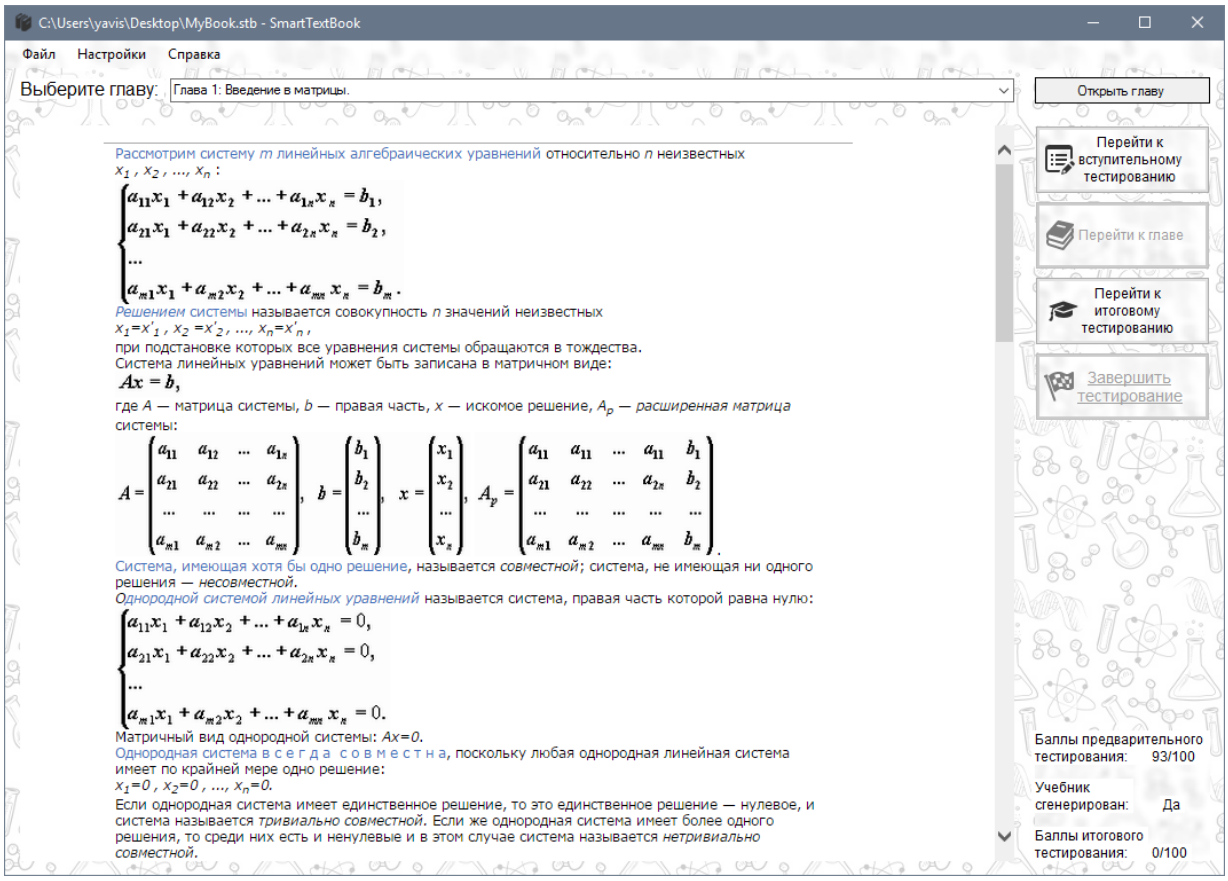

Рисунок 3.4 – Окно с текстом учебника

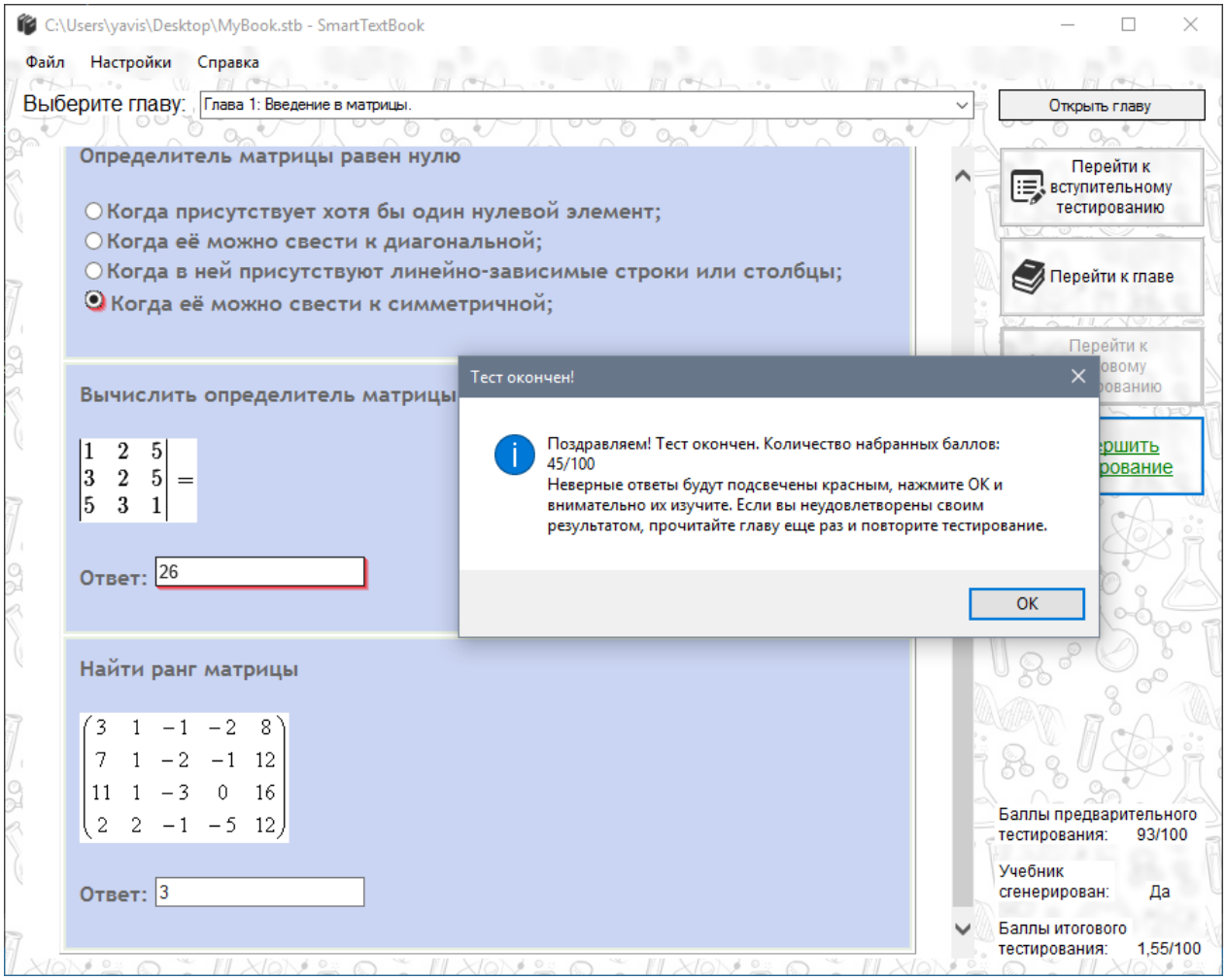

Рисунок 3.5 – Окно финального тестирования

На этом рисунке можно наблюдать неверный ответ, подсвеченный красным цветом.

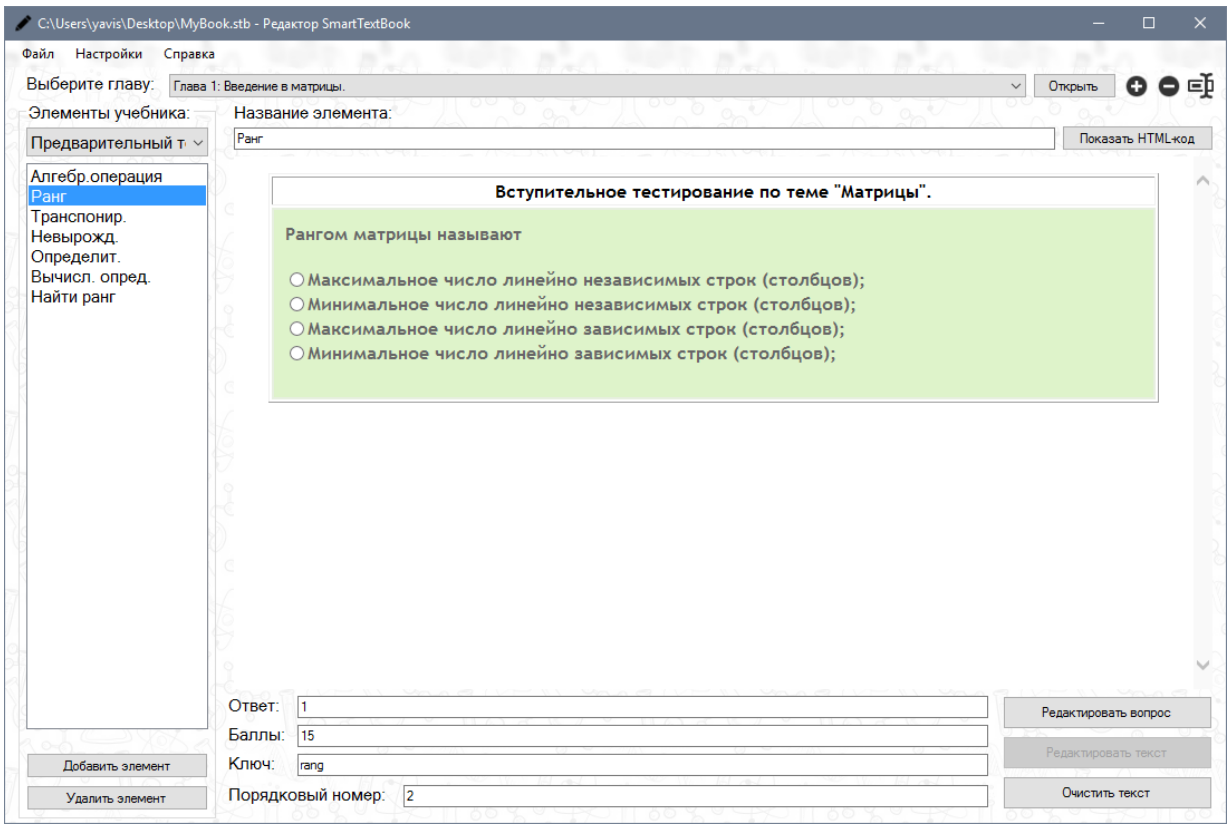

Рисунок 3.6 – Окно создания/редактирования учебника

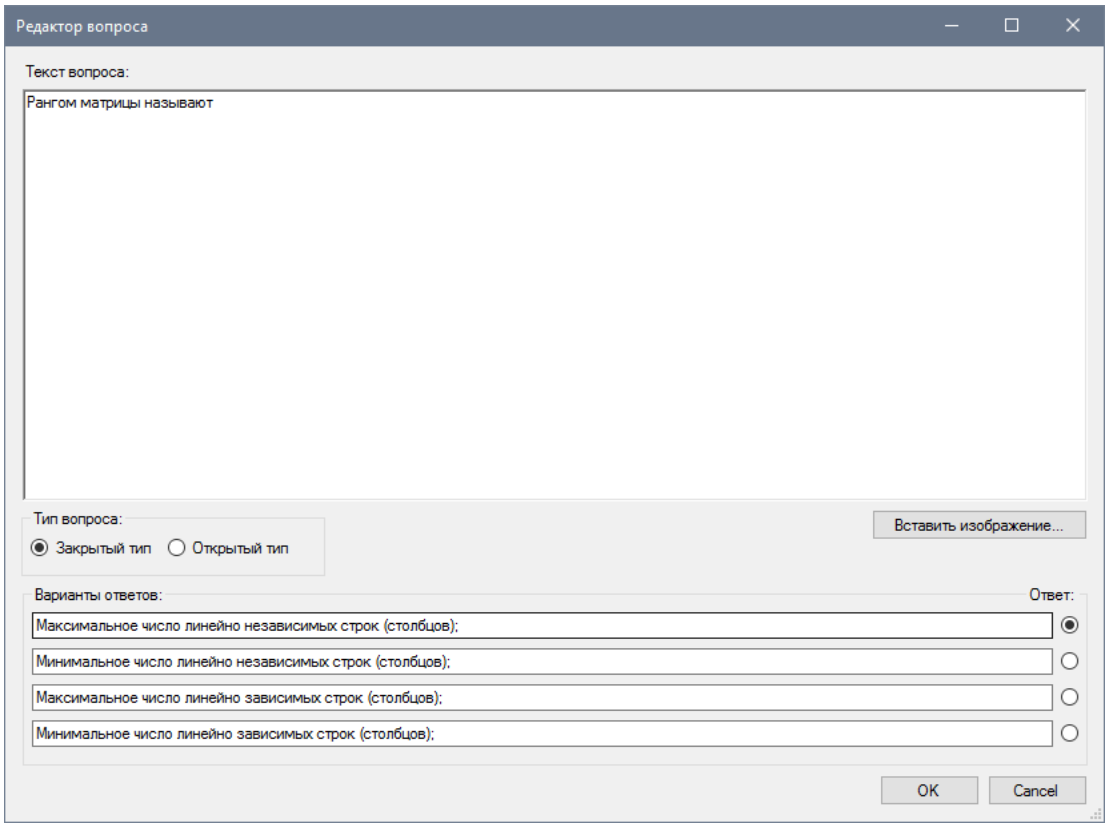

Рисунок 3.7 – Окно редактирования вопроса

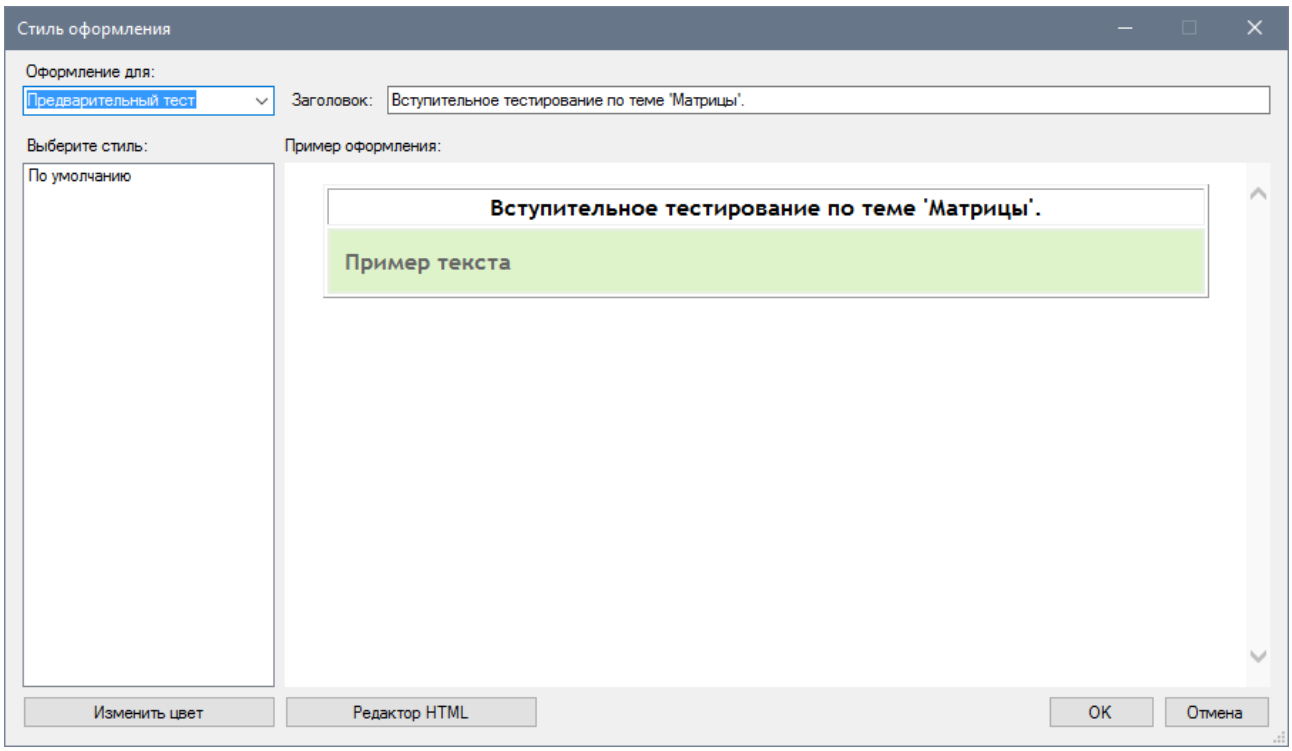

Рисунок 3.8 – Окно редактирования стилей

ЗАКЛЮЧЕНИЕ

1. Спроектировано и разработано ПО, представляющее собой обучающую адаптивную систему.

2. В системе реализованы:

- Возможность адаптации к обучаемому посредством предварительного тестирования;

- Возможность обучаемому самостоятельно контролировать изученные знания посредством итогового тестирования;

- Возможность редактирования и создания собственных учебников;

- Возможность загрузки и хранения учебников в онлайн-облаке.

3. В данный момент идёт процесс внедрения разработанного ПО в качестве обучающего средства в НИИ Медицинских Проблем Севера.

4. Планируется подача заявления на государственную регистрацию системы в качестве программы для ЭВМ в федеральном государственном бюджетном учреждении «Федеральный институт промышленной собственности» (ФИПС).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Быкова, В.В. Проектирование баз данных: теория нормализации в задачах и упражнениях : учеб. пособие / В.В. Быкова. – Красноярск : ИЦ ин-та естеств. и гуманит. наук, 2007. – 106 с.
2. Вирт, Н. Алгоритмы и структуры данных. Новая версия для оберона / Н. Вирт. – Москва : ДМК Пресс, 2010 – 272 с.
3. Осипов, Н.А. Разработка Windows приложений на C# : учеб. пособие / Осипов Н.А. – Санкт-Петербург : НИУ ИТМО, 2012. – 74с.
4. Петцольд, Ч. Программирование для Microsoft Windows на C#: в 2 т. / Ч. Петцольд. – Москва : Издательско-торговый дом «Русская Редакция», 2002. – Т.1. – 576 с.
5. Петцольд, Ч. Программирование для Microsoft Windows на C#: в 2 т. / Ч. Петцольд. – Москва : Издательско-торговый дом «Русская Редакция», 2002. – Т.2. – 624 с.
6. Роббинс, Д. HTML5: карманный справочник / Д. Роббинс. – Москва : ООО "И.Д. Вильямс", 2015. - 192 с.
7. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – Москва : ООО “И.Д. Вильямс”, 2013. – 1312 с.
8. Фицджеральд, М. Регулярные выражения: основы / М. Фицджеральд. – Москва : ООО "И.Д. Вильямс", 2015. – 144 с.
9. Шилдт, Г. C# 4.0: полное руководство / Г. Шилдт. – Москва : ООО "И.Д. Вильямс", 2011. – 1056 с.