МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ЮУрГГПУ»)

ФАКУЛЬТЕТ МАТЕМАТИКИ, ФИЗИКИ, ИНФОРМАТИКИ

КАФЕДРА ИНФОРМАТИКИ, ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И МЕТОДИКИ ОБУЧЕНИЯ ИНФОРМАТИКЕ

Разработка автоматизированной обучающей системы для организации самостоятельной работы учащихся школы

Выпускная квалификационная работа

по направлению 09.03.02 Информационные системы и технологии

Направленность программы бакалавриата

«Информационные технологии в образовании»

Форма обучения очная

Проверка на объем заимствований:
73,55 % авторского текста

Работа рекомендована к защите
рекомендована/не рекомендована

« 11 » мая 2022 г.
зав. кафедрой И, ИТ и МОИ

_____ Рузаков А.А.

Выполнил:
Студент группы ОФ-413-095-4-1
Фатхутдинов Иван Александрович

Научный руководитель:
к.п.н., доцент кафедры ИИТМОИ

_____ Давыдова Н.А.

Челябинск
2023

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ЮУрГГПУ»)

ФАКУЛЬТЕТ МАТЕМАТИКИ, ФИЗИКИ, ИНФОРМАТИКИ

КАФЕДРА ИНФОРМАТИКИ, ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И МЕТОДИКИ ОБУЧЕНИЯ ИНФОРМАТИКЕ

Разработка автоматизированной обучающей системы для организации самостоятельной работы учащихся школы

Выпускная квалификационная работа

по направлению 09.03.02 Информационные системы и технологии

Направленность программы бакалавриата

«Информационные технологии в образовании»

Форма обучения очная

Проверка на объем заимствований:
_____% авторского текста

Работа _____ к защите
рекомендована/не рекомендована

« ___ » _____ 20__ г.
зав. кафедрой И, ИТ и МОИ

_____ Рузаков А.А.

Выполнил:
Студент группы ОФ-413-095-4-1
Фатхутдинов Иван Александрович

Научный руководитель:
к.п.н., доцент кафедры ИИТМОИ

_____ Давыдова Н.А.

Челябинск
2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компьютеры в значительной степени стали неотъемлемой частью нашей жизни. В частности, в области образования, их использование весьма разнообразно.

Компьютерные технологии оказывают глубокое воздействие на сектор образования. Благодаря компьютерам, процесс обучения стал гораздо проще и интереснее, чем раньше. Они обеспечивают быструю обработку данных с очень небольшой вероятностью ошибок в обработке [1, с. 22].

Появление интерактивных систем тестирования напрямую связано с развитием дистанционного обучения. Дистанционное обучение (ДО) – взаимодействие учителя и учащихся между собой на расстоянии, отражающее все присущие учебному процессу компоненты (цели, содержание, методы, организационные формы, средства обучения) и реализуемое специфичными средствами Интернет-технологий или другими средствами, предусматривающими интерактивность [2, с. 17-19].

Целью работы является проектирование и разработка автоматизированной обучающей системы FruOpus. При этом, содержание учебного материала, размещенного в системе, подразумевает расширение и углубление разделов школьной информатики.

Для достижения поставленной цели были поставлены следующие задачи:

1) Изучить организационную структуру и бизнес-процессы Государственного бюджетного учреждения дополнительного профессионального образования «Челябинский институт развития образования».

2) Описать объект автоматизации и сформулировать требования к функционированию разрабатываемой системы.

3) Провести анализ аналогичных программных продуктов.

4) Спроектировать автоматизированную обучающую систему и описать реализацию ее основных функций.

5) Составить руководство пользователя и системного программиста.

6) Провести технико-экономическое обоснование разработанного программного обеспечения.

Структурно работа состоит из введения, трёх глав, заключения и приложений. В первой главе дано описание предметной области, во второй – описан процесс разработки, в третьей приводится документация. В заключении отражены основные выводы по результатам работы, оценка полноты решения поставленных задач.

ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И НАЗНАЧЕНИЕ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

1.1 Характеристика объекта информатизации

1.1.1 Описание ГБУ ДПО «Челябинский институт развития образования»

Государственное бюджетное учреждение дополнительного профессионального образования «Челябинский институт развития образования» было основано 26.01.2005 года.

Информация о ГБУ ДПО «ЧИРО»:

- Адрес юридический: 454091 г. Челябинск, ул. Комсомольская, 20а.
- Адрес фактический: 454091 г. Челябинск, ул. Комсомольская, 20а.
- 454087 г. Челябинск, ул. Блюхера 91.
- Телефон (факс) 8 (351) 217-30-89 (приемная).
- Электронная почта: info@rcokio.ru.
- Сайт: https://rcokio.ru.

Основными видами деятельности Учреждения являются:

1. Проведение научных исследований, опытно-экспериментальной и научно-методической работы.
2. Методическое сопровождение реализации образовательных программ образовательными организациями.
3. Реализация программ повышения квалификации.
4. Реализация программ профессиональной переподготовки.
5. Организационное и технологическое обеспечение проведения единого государственного экзамена в Челябинской области, осуществление функций регионального центра обработки информации.
6. Методическое, информационное, организационное и техническое обеспечение проведения процедуры аттестации педагогических работников.

7. Организация и проведение мероприятий в сфере образования и науки;

8. Проведение видеоконференций.

9. Ресурсное и информационно-технологическое обеспечение образовательных организаций.

10. Транспортное обслуживание Министерства образования и науки Челябинской области.

11. Ведение Интернет-сайта Министерства образования и науки Челябинской области.

12. Редакционно-издательская и типографская деятельность.

13. Информационное и методическое обеспечение деятельности Министерства образования и науки Челябинской области.

14. Методическое, информационное, организационное и техническое обеспечение оценки качества образования в Челябинской области.

15. Осуществление технического исполнения работ, проведение технической актуализации и поддержки пространственных данных геоинформационной системы «Геопортал Челябинской области».

16. Организационно-техническое и научно-методическое сопровождение мониторинга системы образования в Челябинской области.

17. Обеспечение информационной безопасности вычислительных систем, сетей и баз данных образовательных организаций, проведение работ по аттестации объектов информатизации.

18. Методическое, информационное, организационное и техническое обеспечение функционирования информационных систем в сфере образования.

1.1.2 Организационная структура

Организационная структура управления – совокупность специализированных функциональных подразделений, взаимосвязанных в

процессе обоснования, выработки, принятия и реализации управленческих решений. Организационная структура ГБУ ДПО «ЧИРО» представлена на рисунке 1.

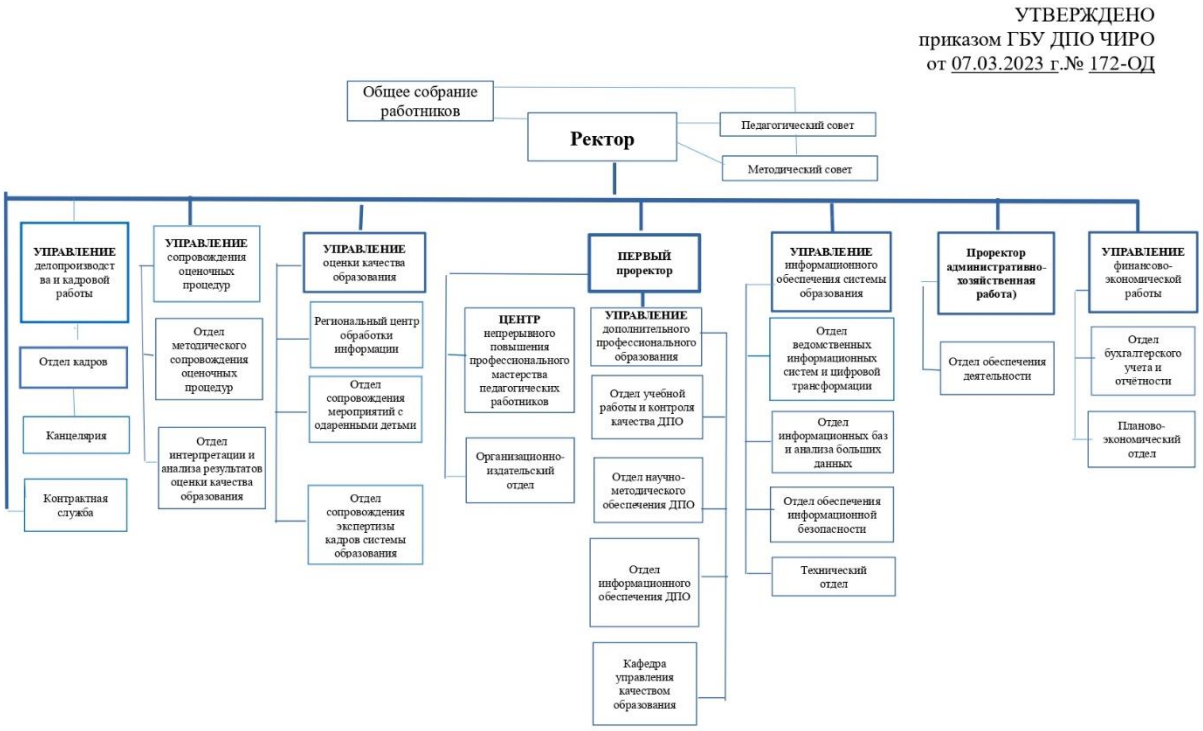

Рисунок 1 – Организационная структура ГБУ ДПО «ЧИРО»

1.1.3 Описание бизнес-процессов

Для описания функциональной модели контроля знаний учащихся была выбрана методология IDEF0 так как она является наиболее удобной для отображения бизнес-процессов. Методология IDEF0 нашла широкое применение благодаря простому отображению информации. Главным компонентом модели являются диаграммы. На них отображаются функции системы в виде прямоугольников, а также связи между ними и внешней средой посредством стрелок. Использование такого малого количества графических примитив позволяют быстро объяснить правила и принципы построения диаграмм IDEF0 людям, незнакомым с данной методологией. Это достоинство позволяет подключить и активизировать деятельность заказчика по описанию бизнес-процессов с использованием формального и

наглядного графического языка. Контекстная диаграмма с описанием процессов организации представлена на рисунке 2.

Рисунок 2 – Бизнес-процесс контроля знаний учеников

Далее, на рисунке 3, представлена декомпозиция контекстной диаграммы.

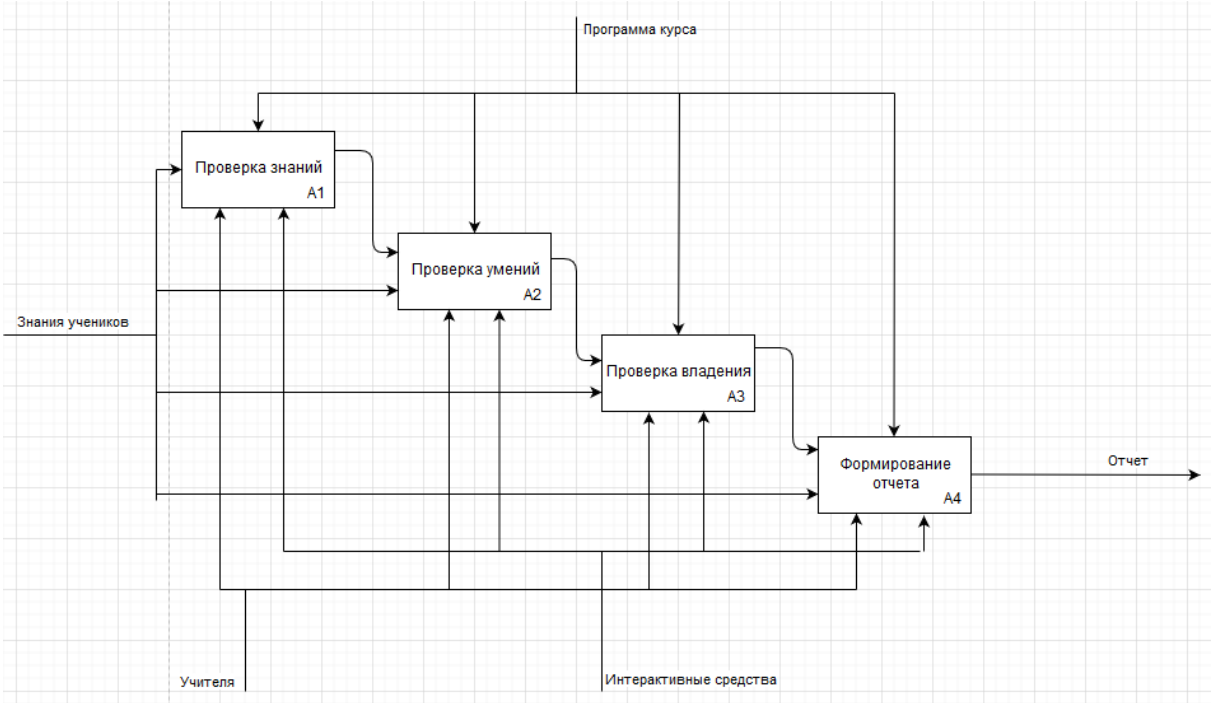

Рисунок 3 – Декомпозиция контекстной диаграммы

Так же были сформированы следующие требования для приложения:

– обучение,

– проверка уровня знаний ученика,
– возможность формирования отчета,
– оценка уровня знаний ученика,
– обнаружение “пробелов” в знаниях.

1.2 Описание проектируемой системы

1.2.1 Назначение системы

Предметом автоматизации является организация самостоятельной работы учащихся, углубленное изучение информатики в старших классах. Система предназначена для углубленного изучения некоторых разделов информатики и контроля полученных знаний.

1.2.2 Цели создания системы

Целями создания FRUOPUS является углубление и расширение знаний школьников по разделам информатики.

1.2.3 Требования к структуре и функционированию системы

1.2.3.1 Перечень подсистем

FRUOPUS состоит из подсистемы лекций и подсистемы тестов.

Подсистема лекций предназначена для предоставления обучающемуся структурированного по темам и разделам лекционного материала по предмету обучения. Подсистема теории загружает информацию с файлов лекций и базы данных.

Подсистема тестов предназначена для проверки полученных знаний и рекомендации обучающемуся подходящей лекции для повторного прочтения в том случае, если он ошибся. Информацию подсистема тестирования загружает из базы данных [5, с. 24-25].

Отдельно следует выделить также подсистему регистрации и авторизации.

Структурная схема системы представлена на рисунке 4.

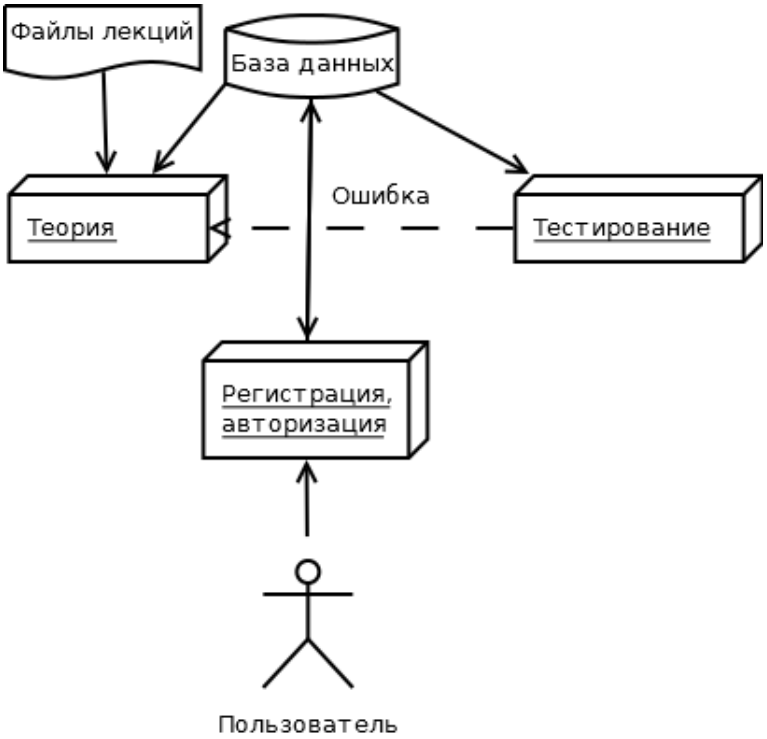

Рисунок 4 – Структурная схема FRUOPUS

1.2.3.2 Требования к способам и средствам связи для информационного обмена между компонентами системы

Связь подсистем осуществляется через базу данных.

1.2.3.3 Требования к совместимости

Совместимость с операционной системой Windows 7 и новее.

1.2.3.4 Требования к режимам функционирования системы

FRUOPUS должна поддерживать следующие режимы функционирования:

1) регистрация и авторизация;
2) изучение материалов лекций;

3) прохождение тестов.

1.2.3.5 Требования по диагностированию системы

Проверка целостности данных и нарушений проводится по мере необходимости. Проверка программного и аппаратного обеспечения проводится по мере необходимости.

1.2.3.6 Перспективы развития, модернизации системы в целом

Будущие версии FRUOPUS можно усовершенствовать, добавляя новые возможности:

1) учёт ответов пользователей в базе данных;

2) возможность администратору просматривать ответы пользователей;

3) возможность администратору добавлять новые разделы, лекции и задания;

4) использования большего типа задний.

1.2.4 Требования к функциям

Система должна быть многопользовательской с возможностью регистрации и авторизации.

Зарегистрированный пользователь должен иметь возможность авторизации. Для авторизации пользователь должен ввести логин и пароль и, в случае неправильного ввода, получить сообщение об ошибке.

Незарегистрированный пользователь должен иметь возможность регистрации. Для регистрации пользователь должен ввести логин и пароль, а также подтвердить введённый пароль, и в случае неправильного ввода или ввода уже занятого логина должна выдавать сообщение об ошибке.

Логин должен быть уникальной непустой строкой, состоящей из алфавитно-цифровых символов, между которыми может находиться по одному пробелу.

Пароль должен быть непустой строкой любого размера.

После успешной регистрации или авторизации пользователю должна стать доступна главная форма программы с двумя вкладками: теория и тестирование.

Вкладка «Теория» должна состоять из списка разделов, списка тем и окна вывода теоретического материала.

Когда пользователь выбирает в списке определённый раздел, система показывает ему список тем из этого раздела.

Когда пользователь выбирает в списке определённую тему, система показывает ему в специальном окне материал лекции по выбранной теме.

Вкладка «Тестирование» должна иметь выпадающий список для выбора уровня сложности (1 или 2) и кнопку, позволяющую запустить или перезапустить тест с заданиями выбранного уровня сложности.

Максимальная оценка за тест должна составлять 10 баллов. Тест уровня 1 состоит из 10 вопросов по 1 балла. Тест уровня 2 состоит из 5 вопросов по 2 балла.

При нажатии кнопки запуска теста счётчик заданий и счётчик баллов обнуляются, а пользователю показывается первое случайное задание с вариантами ответа на него. Пока пользователь не даст ответ, переход к следующему заданию будет для него недоступен. Если пользователь дал ответ, ему выводится результат и предлагается перейти к следующему заданию.

Каждое задание должно состоять из нескольких вариантов ответа, среди которых пользователю предлагается выбрать один правильный.

В случае правильного ответа пользователю выводится сообщение и начисляются баллы в зависимости от уровня сложности.

В случае неправильного ответа пользователю выдаётся сообщение с информацией об ошибке и ссылка на лекцию, с которой ему следует ознакомиться для исправления сделанной ошибки.

При переходе по ссылке открывается вкладка «Теория», а в специальной окошке отображается материал по лекции, которую ему надо изучить повторно.

Когда все задания выполнены, пользователю выводится итоговая оценка (от 0 до 10).

1.3 Сравнительный анализ аналогичных программных продуктов

Попробуем выбрать несколько существующих систем тестирования и провести их краткий анализ, проследить историю перехода интерактивного обучения от offline к online, выбрать лучшую. Для анализа выберем наиболее релевантные и широко используемые автоматизированные системы тестирования, такие как PikaTest, UniTest, Indigo и Moodle.

PikaTest – это бесплатная программа для создания и проведения двухуровневых тестов с неограниченным количеством вопросов, по типу ЕГЭ. Она интересна в первую очередь тем, что является offline системой тестирования. В настоящее время это встречается крайне редко [7, с. 4].

PikaTest появилась в начале 2012 года. С помощью этой программы можно создать полноценный тест, с неограниченным количеством вопросов. Тест может содержать аудио/видео - файлы, а также таблицы и изображения. Имеется возможность добавления вопроса с вариантами ответов и без них, а также указание «веса» каждого отдельного вопроса. Программа позволяет создавать тесты с ограниченным временем прохождения. Файлы тестов сохраняются в формате *.pikatest, который воспринимается только программой. Подробный отчет о тестировании сохраняется в формате *.txt и доступен для просмотра [7, с. 5].

Прохождение тестирования осуществляется по следующей схеме:

– регистрация пользователя;

– настройки прохождения тестирования;

– вывод подробной статистики о результатах тестирования с детализацией до каждого вопроса;

– сохранение отчета о результатах тестирования;

– отправка отчета по электронной почте.

Основные преимущества:

– система независима и имеет небольшой размер (всего 1761 Kb);

– система является бесплатной;

– программа легка в обращении, и не требует каких-то специальных знаний;

– совместима со всеми распространенными ОС (windows 7/8/ 10)

Недостатки:

– система работает в режиме offline, при этом справка по программе доступна только в интернете;

– отсутствие четкого разделения администратор-пользователь;

– незащищенность данных;

– отсутствие множества полезных функций.

UniTest – это комплексное программное решение для проведения компьютерного тестирования, функционально реализованное по CAM-технологии [8, с. 8] (CAM – с англ. Computer-Aided Manufacturing). Первые версии программы появились еще в 2007 году.

Программа предназначена для формирования банка тестовых заданий и организации процесса проверки знаний [8, с. 9]. Позволяет организовать процесс контроля знаний с помощью компьютерного тестирования в сети с использованием транспортного протокола TCP/IP, а также локального тестирования [6, с. 12-14].

Система UniTest программно реализована с применением технологий Microsoft.NET 3.0, уникальных алгоритмов параллельной обработки информации и передовых средств криптографической защиты [7].

Система является бесплатной и доступна всем пользователям на официальном сайте продукта.

Основные преимущества:

– работа с пакетом комфортна и понятна простому пользователю;

– удобный интерфейс, самые полные тесты занимаю очень мало места (500-700 Kb памяти);

– поддержка всех основных и множества дополнительных типов вопросов;

– высокий уровень защиты данных (все данные шифруются BlowFish 448 бит);

– возможность проведения тестирования как локально, так и по сети;

– поддержка большого числа языков;

– огромное количество полезных функций.

Недостатки: несовместима с последними версиями ОС (Windows 10/11); высокие требования к техническим средствам; назначение логина и пароля, для аутентификации преподавателей и студентов, непосредственно «Администратором» (так сказать «вручную»).

Indigo – представляет собой мультифункциональный комплекс программного обеспечения, позволяющий автоматизировать процесс проведения тестирования и обработки результатов [4, с. 35-37]. Продукт был разработан в 2010 году.

Основные преимущества: простая установка системы; доступный интерфейс пользователя; продукт совместим со всеми ОС семейства Windows (XP/2003/Vista /7/8); поддержка всех распространенных браузеров; централизованное хранение данных и Web-интерфейс пользователей;

иерархическая группировка тестов и пользователей (правила тестирования); широкие возможности конструктора тестов.

Недостатки: система является платной, отсутствие разделения администратор-преподаватель (не всегда преподаватель имеет навыки работы с подобного рода системами); относительно большой объем потребляемой памяти; высокие требования к оборудованию.

Moodle – это система управления содержимым сайта (Content Management System CMS), специально разработанная для создания онлайн-курсов преподавателями. Такие e-learning системы часто называются системами управления обучением (Learning Management Systems – LMS) или виртуальными образовательными средами (Virtual Learning Environments – VLE) [4, с. 39]. Moodle написана на языке программирования PHP профессором из Австралии Мартином Дунгиамосом и переведена на несколько десятков языков и используется для обучения более чем в ста пятидесяти странах мира.

Основные преимущества: полный набор необходимых функций; открытый исходный код продукта (что позволяет добавить все необходимые элементы); система Moodle универсальна в плане требований (любая ОС, установленный модуль PHP и одна из СУБД); все виды тестов (включая написание эссе).

Недостатки: система тестирования является частью большого программного продукта; обслуживание предоставляется за отдельную плату.

Выводы по главе 1

В данной главе были сформулированы требования к FRUOPUS, определён набор функций, которыми она должна обладать; проанализирован рынок существующих решений, технология их реализации и недостатки, а также рассмотрен и охарактеризован ГБУ ДПО «ЧИРО».

ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ FruOpus

2.1 Выбор архитектуры информационной среды и инструментальных средств реализации

Для реализации FRUOPUS выбран язык программирования C# и среда разработки Visual Studio 2019. Их преимуществами является абстрагирование от аппаратной части компьютера с возможностью сконцентрироваться исключительно на прикладных задачах, объёмная стандартная библиотека со встроенной документацией, лёгкость проектирования пользовательского интерфейса, хорошо развитая система подсказок в процессе кодирования, простота работы с базами данных.

Для создания базы данных выберем систему Microsoft Access. Её преимуществами является возможность легко создавать, связывать и заполнять таблицы в режиме визуального проектирования. Также плюсом Access является отсутствие необходимости устанавливать сервер баз данных, что значительно упрощает переносимость базы данных с одного устройства на другое.

2.2 Описание модели данных

2.2.1 Проектирование базы данных

В рассматриваемой предметной области можем выделить следующие сущности: пользователь (свойства: логин, пароль), раздел (свойство: название), тема (свойства: название, раздел), задание (свойства: текст, тема, уровень, номер правильного ответа), вариант ответа (свойства: текст, задание) [3, с. 34]. Под каждую сущность создадим отдельную таблицу.

Описание таблицы «Пользователи» приведено на рисунке 5.

Имя поля	Тип данных	
Логин	Короткий текст	Логин пользователя
Пароль	Короткий текст	Пароль пользователя
Тип	Числовой	Код типа пользователя

Рисунок 5 – Структура таблицы «Пользователи»

Описание таблицы «Разделы» приведено на рисунке 6.

Имя поля	Тип данных	
Код	Счетчик	Код раздела
Название	Короткий текст	Название раздела

Рисунок 6 – Структура таблицы «Разделы»

Описание таблицы «Темы» приведено на рисунке 7.

Имя поля	Тип данных	
Код	Счетчик	Код темы
Название	Короткий текст	Название темы
Раздел	Числовой	Код раздела

Рисунок 7 – Структура таблицы «Темы»

Описание таблицы «Задания» приведено на рисунке 8.

Имя поля	Тип данных	
Текст	Короткий текст	Текст задания
Тема	Числовой	Код темы
Уровень	Числовой	Уровень сложности
Ответ	Числовой	Номер правильного варианта

Рисунок 8 – Структура таблицы «Задания»

Описание таблицы «Варианты» приведено на рисунке 9.

Имя поля	Тип данных	
Код	Счетчик	Код варианта
Текст	Короткий текст	Текст варианта
Задание	Числовой	Код задания

Рисунок 9 – Структура таблицы «Варианты»

Между разделами и темами существует связь «один ко многим»: в каждом разделе есть одна и более тем, но каждая тема относится к одному и только одному разделу.

Между темами и заданиями существует связь «один ко многим»: по каждой теме есть одно и более заданий, но каждое задание относится к одной и только одной теме.

Между заданиями и вариантами существует связь «один ко многим»: для каждого задания есть несколько вариантов ответа, но каждый вариант относится к одному и только одному заданию.

Таблица пользователей стоит отдельно и не связана с другими таблицами.

На рисунке 10 показана схема разработанной базы данных.

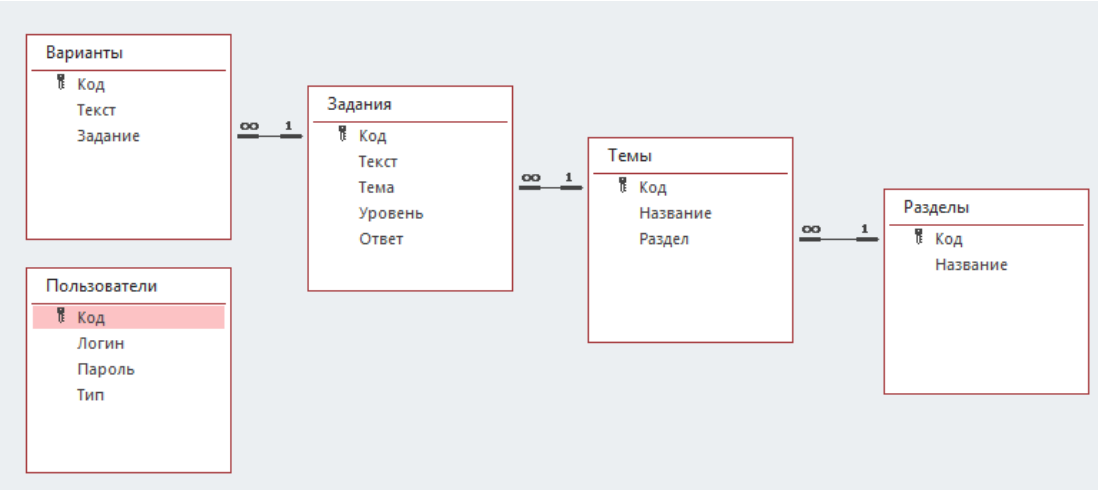

Рисунок 10 – Схема базы данных

База данных находится в первой нормальной форме, поскольку каждая таблица имеет первичный ключ и все ячейки таблиц имеют простую структуру.

База данных находится во второй нормальной форме, потому что в ней нет таблиц со сложными первичными ключами.

База данных находится в третьей нормальной форме, так как ни в одной таблицы нет зависимости от неключевых полей.

2.2.2 Разработка иерархии классов

Для всех сущностей, описанных в предыдущем пункте, создадим классы под платформу .NET Framework на языке программирования C#.

2.2.2.1 Класс Section

Для работы с тематическими разделами создадим класс Section.

Список тем, относящихся к данному разделу, будем хранить в закрытом поле themes типа List<Theme>. Доступ для чтения к этому полю будет осуществляться через открытое свойство Themes такого же типа.

Код раздела будем хранить в свойстве ID типа int, а его название – в свойстве Name типа string.

Переопределим унаследованный от встроенного класса System.Object метод ToString так, чтобы он возвращал свойство Name.

Создадим открытый статический метод SelectAll, который будет выбирать из базы данных все тематические разделы и возвращать список типа List<Section>.

Код класса разместим в файле Section.cs приложение 1.

2.2.2.2 Класс Theme

Для работы с темами создадим класс Theme. Код темы будем хранить в свойстве ID типа int, её название – в свойстве Name типа string, а ссылку на секцию, к которой тема относится, – в свойстве Section.

Переопределим унаследованный от встроенного класса System.Object метод ToString так, чтобы он возвращал свойство Name.

Определим открытый статический метод Select, который будет отбирать из базы данных и возвращать тему с заданным кодом, передаваемом ему в качестве параметра id типа int.

Определим открытый статический метод SelectAll, который будет отбирать из базы данных все темы определённого раздела и возвращать список List<Theme>. Раздел передаётся методу в качестве параметра section типа Section.

Код класса разместим в файле Theme.cs приложение 2.

2.2.2.3 Класс Task

Для работы с заданиями создадим класс Task.

Код темы, к которой относится задание, будем хранить в закрытом поле themeID типа int, а ссылку на эту тему – в закрытом поле theme типа Theme. Доступ к этому полю для чтения будем осуществлять с помощью открытого свойства Theme.

Список вариантов ответа на задание будем хранить в закрытом поле variants типа List<Variant>. Доступ для чтения к этому полю будет осуществляться через открытое свойство Variants такого же типа.

Код задания будем хранить в свойстве ID типа int, его текст – в свойстве Text типа string, уровень сложности – в свойстве Level типа byte, номер правильного ответа – в свойстве Answer типа byte.

Переопределим унаследованный от встроенного класса System.Object метод ToString так, чтобы он возвращал свойство Text.

Создадим открытый статический метод SelectAll, который будет выбирать из базы данных все тематические разделы и возвращать список типа List<Section>.

Определим открытый статический метод Select, который будет отбирать из базы данных все задания определённого уровня сложности и возвращать их список типа List<Task>. Уровень сложности передаётся ему в качестве параметра level типа byte.

Код класса разместим в файле Task.cs приложение 3.

2.2.2.4 Класс Variant

Для работы с вариантами ответа создадим класс Variant. Код варианта ответа будем хранить в свойстве ID типа int, его текст – в свойстве Text типа string, а ссылку на задание, к которому он относится – в свойстве Task.

Переопределим унаследованный от встроенного класса System.Object метод ToString так, чтобы он возвращал свойство Text.

Определим открытый статический метод Select, который будет отбирать из базы данных все варианты ответа на определённое задание и

возвращать их список типа List<Variant>. Ссылка на задание передаётся ему в качестве параметра task типа Task.

Код класса разместим в файле Variant.cs приложение 4.

2.2.2.5 Класс User

Для работы с пользователями создадим класс User.

Логин пользователя будем хранить в закрытом поле login типа string. Доступ к этому полю будем осуществлять через открытое свойство Login типа string. В этом свойстве будет проверять, чтобы логин был непустым, не содержал лишних пробелов, состоял только из алфавитно-цифровых символов, возможно разделённых пробелами.

Пароль пользователя будем хранить в закрытом поле password типа string. Доступ к этому полю будем осуществлять через открытое свойство Password типа string. В этом свойстве будем проверять, чтобы пароль был непустым.

Код пользователя будем хранить в открытом свойстве ID типа int.

Является ли пользователь администратором системы, будем определять открытое свойство IsAdmin типа bool.

Для проверки того, присутствует ли пользователь в базе данных, определим открытый метод Exists, возвращающий true, если пользователь присутствует в базе, и false, если нет. Метод имеет параметр withPass типа bool со значением по умолчанию true. Если он имеет значение false, присутствие пользователя в базе данных определяется только по логину, а если true – и по логину. И по паролю, что необходимо для авторизации.

Для регистрации пользователя в базе данных создадим открытый метод Register.

Код класса разместим в файле User.cs приложение 5.

2.2.2.6 Класс Test

Создадим класс Test, организующий процесс прохождения теста. Данный класс является вспомогательным. Его объекты не хранятся в базе данных. Список заданий теста будем хранить в закрытом поле tasks типа List<Task>, а количество заданий теста – в поле count типа byte.

Метод загрузки заданий теста будем вызывать в конструкторе класса. Там же присвоим значение полю count. Параметр level типа byte – номер уровня сложности заданий теста.

Текущее задание теста возвращает открытое свойство CurrentTask типа Task.

Номер текущего задания в тесте возвращает открытое свойство CurrentTaskNumber типа int.

Загрузку заданий теста будет осуществлять закрытый метод LoasTasks. Метод принимает два параметра типа byte: level – уровень сложности, count – количество загружаемых заданий.

Открытый метод Answer вызывается при ответе на текущее задание теста.

Код класса разместим в файле Test.cs приложение 6.

2.2.2.7 Диаграмма классов

Разработанный классы разместим на диаграмме (рисунок 11).

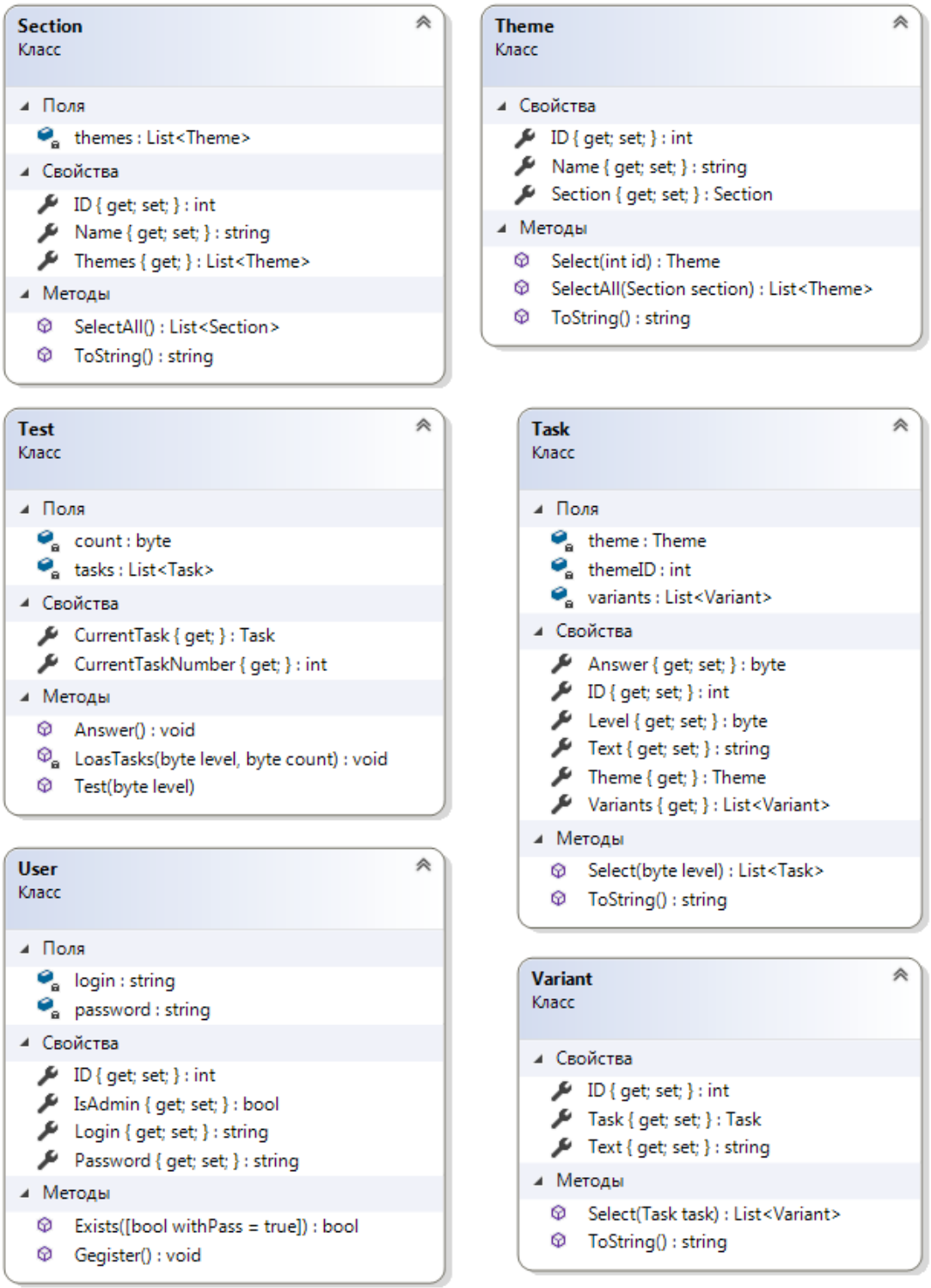

Рисунок 11 – Диаграмма классов

2.3 Описание реализации основных функций системы

2.3.1 Регистрация и авторизация

– Для регистрации и авторизации пользователей создадим специальную форму AuthForm и разместим на ней элементы управления:
– кнопка BtnLogin типа Button с текстом «Войти» - запускает функцию авторизации и регистрации;
– поле TextConfirm типа TextBox – для подтверждения пароля;
– метка label3 типа Label с надписью «подтвердить пароль» – пояснение к полю TextConfirm;
– поле TextPassword типа TextBox – для ввода пароля;
– метка label2 типа Label с надписью «Пароль» – пояснение к полю TextPassword;
– поле TextLogin типа TextBox – для ввода логина;
– метка label1 типа Label с надписью «Логин» – пояснение к полю TextLogin;
– флажок CheckReg типа CheckBox с надписью: «Зарегистрироваться» для переключения между режимами авторизации и регистрации;
– компонент Error типа ErrorProvider – для пометки неправильно заполненных полей.

Структура формы показана на рисунке 12, а внешний вид – на рисунке 13.

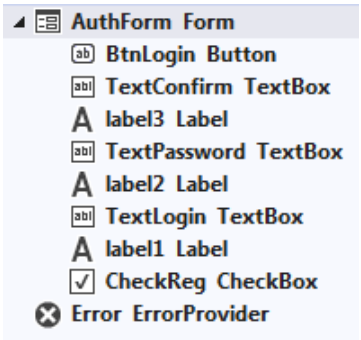

Рисунок 12 – Структура формы AuthForm

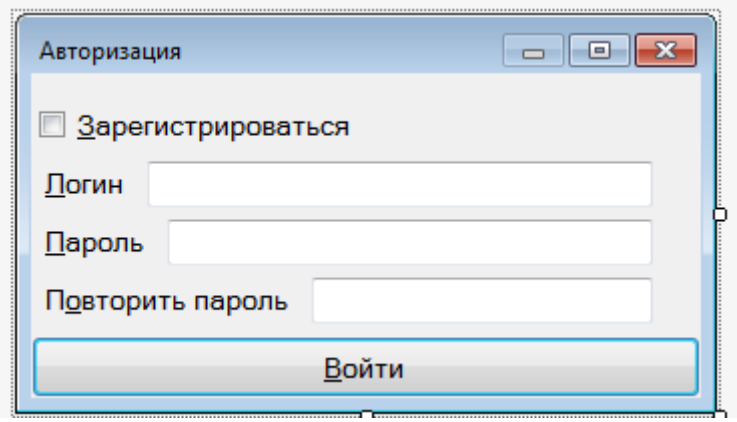

Рисунок 13 – Внешний вид формы AuthForm

При заполнении этой формы возможны ошибки. Для их обработки создадим на базе класса System.Exception класс InputException. В этом классе определим свойство Ctrl типа Control, что ссылается на элемент управление, в котором произошла ошибка.

Конструктор класса принимает два параметра: параметр message типа string – текст сообщения об ошибке, параметр control типа Control – ссылка на элемент управления, в котором произошла ошибка.

Код класса содержится в файле InputException.cs, листинг которого размещён в приложении 7.

Обработчики событий элементов управления формы авторизации и регистрации являются методами класса AuthForm. Метод BtnLogin_Click обрабатывает события Click кнопки BtnLogin. Вначале происходит сброс ошибок и создание объекта текущего пользователя. Далее происходит считывание логина и пароля, и, в случае неправильного ввода, выбрасывается исключение.

Если включён флажок регистрации, проверим, подтверждён ли пароль, а также занят ли логин, и, если нет, убираем исключение. В случае корректности ввода, выполняем регистрацию, и если не будет ошибок, то регистрация считается успешной.

При выключенном флажке регистрации проверяем существование пользователя с таким логином и паролем. Если он существует, авторизация считается успешной, а если нет – убираем исключения.

Схема вышеописанного алгоритма показана на рисунке 14.

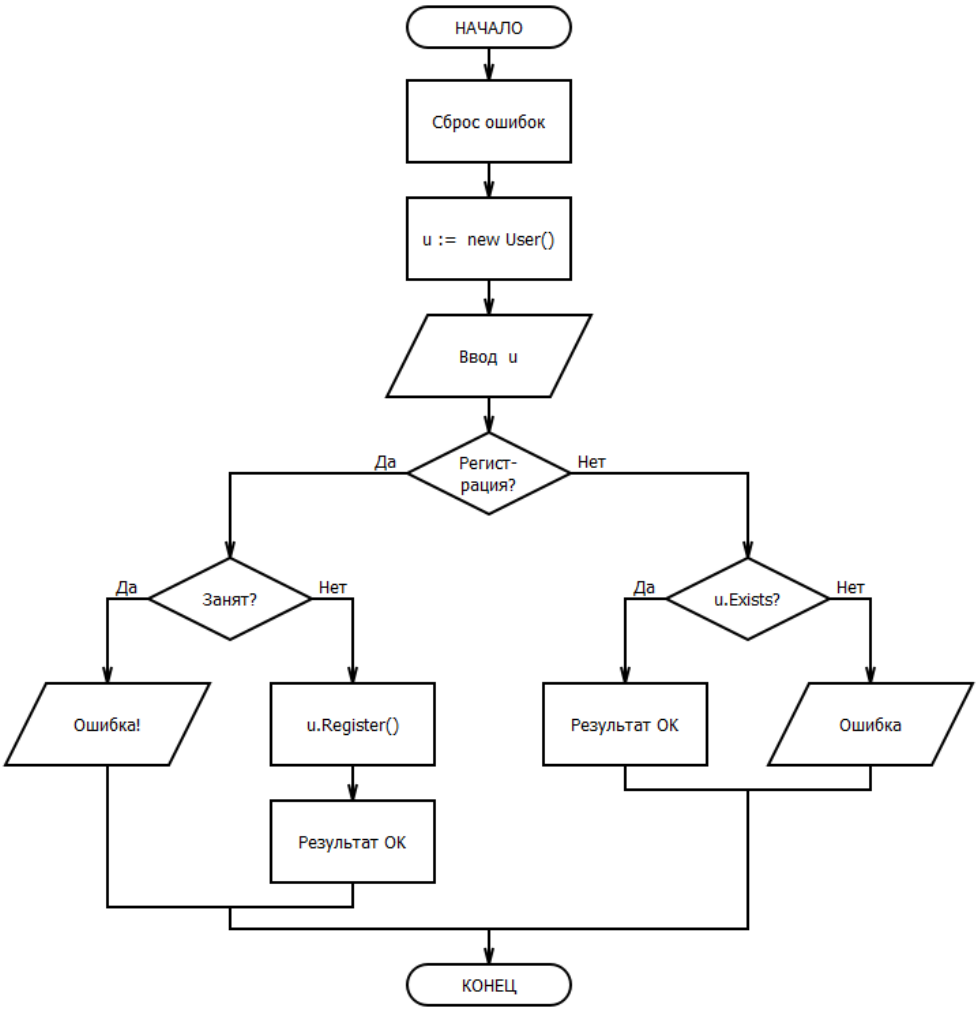

Рисунок 14 – Алгоритм авторизации и регистрации

Метод CheckReg_CheckedChanged является обработчиком события CheckedChanged флажка CheckReg. Он меняет надпись на кнопке и доступность поля подтверждения пароля в зависимости от того, включён флажок или нет.

Описанные здесь обработчики размещены в файле AuthForm.cs, листинг которого дан в приложении 8.

2.3.2 Главная форма

На главной форме, которая и будет выполнять основные функции программы, разместим следующие элементы управления:

– панель вкладок Tabs типа TabControl – для разделения теории и тестирования;

– вкладка tabPage1 типа TabPage с надписью «Теория» – содержит элементы управления для изучения теоретического материала;

– контейнер splitContainer1 типа SplitContainer – для разделения содержимого вкладки «Теория» на меню и окно просмотра лекции с возможностью изменения размера;

– панель groupBox1 типа GroupBox с надписью «Разделы» – для размещения списка тематических разделов;

– список ListSections типа ListBox – для вывода списка тематических разделов с возможностью выбрать интересующий раздел;

– панель groupBox2 типа GroupBox с надписью «Темы» – для размещения списка тем;

– список ListThemes типа ListBox – для вывода списка тем с возможностью выбрать интересующую тему;

– веб-браузер Browser типа WebBrowser – для отображения лекционного материала;

– вкладка tabPage2 типа TabPage с надписью «Тестирование» – содержит элементы управления для прохождения тестов;

– контейнер splitContainer2 типа SplitContainer – для разделения текста задания и вариантов ответа на него с возможностью изменения размера;

– текстовая область RichTask типа RichTextBox – для отображения текста задания;

– список ListVariants типа ListBox – для вывода списка вариантов ответа с возможностью выбрать правильный;

– панель panel1 типа Panel – содержит элементы управления для прохождения тестов;

– ссылка Link типа LinkLabel – для перехода на лекцию с теоретическим материалом по неправильно выполненному заданию;

– метка LabelResult типа Label – для вывода результата – правильности/неправильности ответа;

– кнопка BtnNext типа Button с надписью «Дальше» – для запуска метода перехода к следующему заданию;

– кнопка BtnAnswer типа Button с надписью «Ответить» – для запуска метода ответа на задание;

– меню tools типа ToolStrip – содержит элементы запуска/перезапуска теста с заданным уровнем сложности;

– выпадающий список CmbLevel типа ToolStripComboBox – содержит уровни сложности с возможностью выбрать интересующий;

– кнопка BtnRestart типа ToolStripButton – для запуска метода начала нового теста;

– строка состояния statusStrip1 типа StatusStrip;

– метка Status типа ToolStripStatusLabel – для вывода информационных сообщений.

Иерархическая структура главной формы, порядок размещения и вложенности её элементов показаны на рисунке 15.

Обработчики событий вышеописанных элементов управления являются членами класса MainForm и размещены в файле MainForm.cs, код которого представлен в приложении 9.

Ссылку на текущий тест определим как поле test типа Test.

Оценку за тест определим как поле degree типа byte.

В конструкторе класса будем проводить первичную настройку главной формы, загрузку и вывод списка тематических разделов.

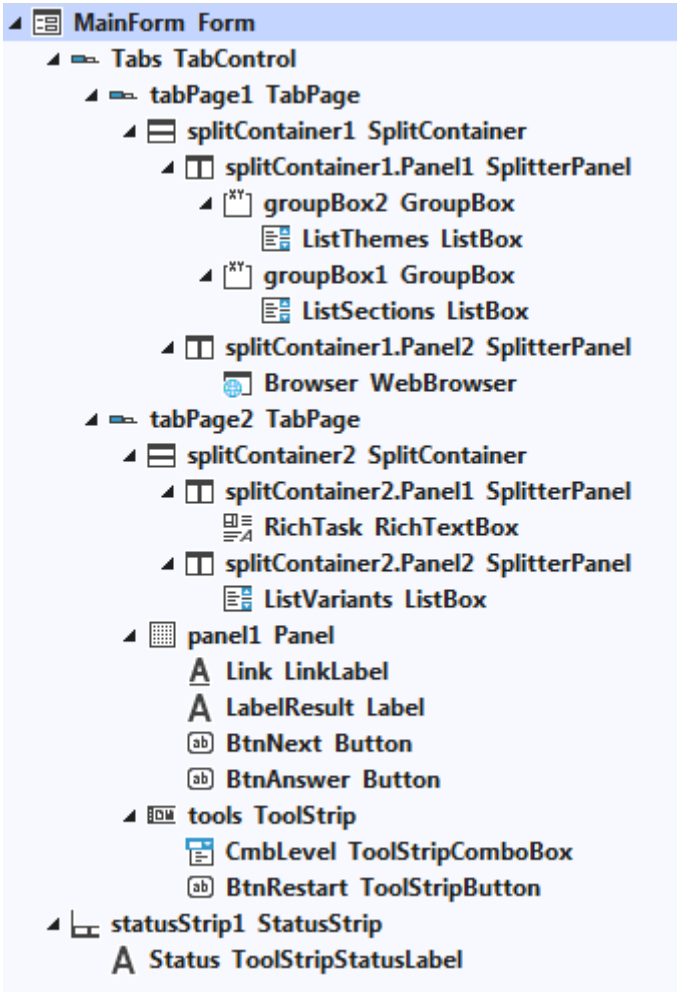

Рисунок 15 – Структура главной формы

За вывод текущего задания отвечает метод ShowTask. Вначале происходит сброс свойств, которые возможно были изменены в ходе ответа на предыдущее задание. Если ссылка на текущее задание пустая, отключаем кнопку ответа, обнуляем список вариантов ответа и выводим полученную за тест оценку, а если нет – выводим тест задания, варианты ответа на него и делаем кнопку ответа доступной.

Метод listSections_SelectedIndexChanged является обработчиком события SelectedIndexChanged списка listSections. Внём осуществляются действия, которые следует произвести после выбора пользователем определённого раздела. Если раздел не выбран, список тем очищается, а если выбран – в данный список загружаются темы этого раздела.

Метод ListThemes_SelectedIndexChanged является обработчиком события SelectedIndexChanged списка ListThemes. Когда пользователь

выбирает в этом списке определённую тему, данный метод загружает в браузер страницу с лекционным материалом по ней.

Метод BtnAnswer_Click является обработчиком события Click кнопки BtnAnswer, срабатывающий при подтверждении пользователем своего ответа. Для текущего теста вызывается метод Answer, кнопка BtnAnswer отключается, а кнопка BtnNext делается доступной. Если пользователь выбрал правильный ответ, ему выводится сообщение и начисляется балл; если неправильный – пользователю выводится сообщение и ссылка на лекцию по теме текущего задания.

Метод BtnNext_Click является обработчиком события Click кнопки BtnNext, единственное действие которого – вызов метода ShowTask, отображающего текущее задание.

Метод Link_LinkClicked является обработчиком события LinkClicked ссылки Link. Данный метод обнуляет список тем, загружает в браузер страницу с лекционным материалом по текущему заданию и активирует вкладку tabePage1.

Метод ReloadTest является обработчиком события Click кнопки BtnRestart. Данный метод создаёт новый тест с выбранным уровнем сложности, обнуляет оценку пользователя и выводит первое задание, вызывая метод ShowTask.

Алгоритм прохождения теста примерно выглядит следующим образом. Начальная оценка degree = 0. Пользователь выбирает уровень сложности n и кликает кнопку запуска теста. Количество заданий теста равно 10/n. Для каждого задания выводится его текст и варианты ответа. Пользователь выбирает ответ. Если ответ верный, пользователю начисляется n баллов, если неправильный – выводится ссылка на лекцию по текущему заданию. В конце теста выводится оценка degree.

Схема данного алгоритма показана на рисунке 16.

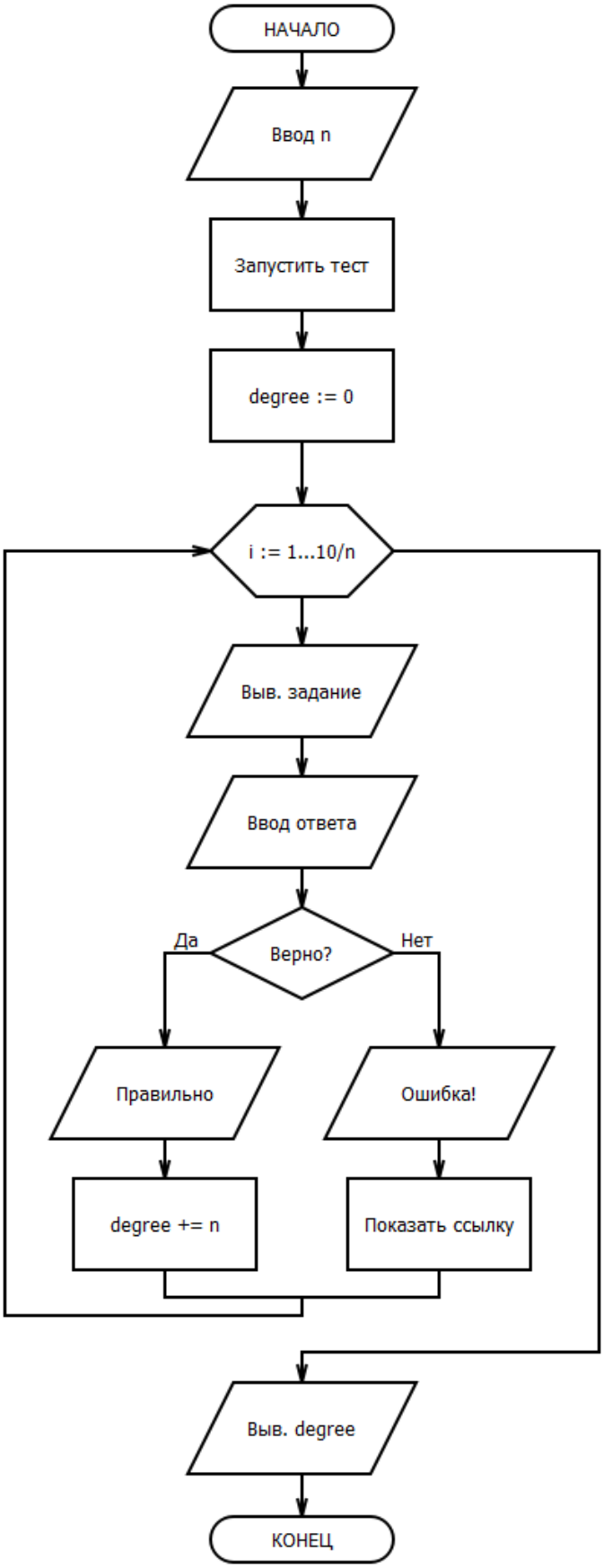

Рисунок 16 – Алгоритм прохождения теста

2.3.3 Главный класс

Главным в разрабатываемой FRUOPUS является статический класс Program. Закрытое поле ConnectString типа string – это строка подключения к базе данных. Закрытое поле Connection типа OleDbConnection – это ссылка на подключение к базе данных. Открытое поле Cmd типа OleDbCommand – это ссылка на объект, отправляющий запросы к базе данных. Ссылка на текущего пользователя хранится в статическом свойстве User.

Точкой входа в программу является метод Main. В нём выполняются подготовительные действия, создаётся и открывается подключение к базе данных, создаётся объект для отправления запросов к базе данных, главная форма и запускается приложение с главной формой.

Код данного класса размещён в файле Program.cs, листинг которого представлен в приложении 10.

Выводы по главе 2

В данной главе был определён список сущностей и заданной предметной области, выяснено, какими свойствами они обладают и как связаны между собой. На основании этой информации в пакете Microsoft Access создана реляционная база данных, заполненная текстами заданий и вариантов ответа. С помощью среды Visual Studio 2019 на языке программирования C# разработано приложение Windows Forms, использующее эту базу данных и выполняющее необходимые функции.

ГЛАВА 3. ДОКУМЕНТИРОВАНИЕ ГОТОВОГО ПРОГРАММНОГО ПРОДУКТА

3.1 Испытание системы

В момент запуска программы появляется окно авторизации (рисунок 17).

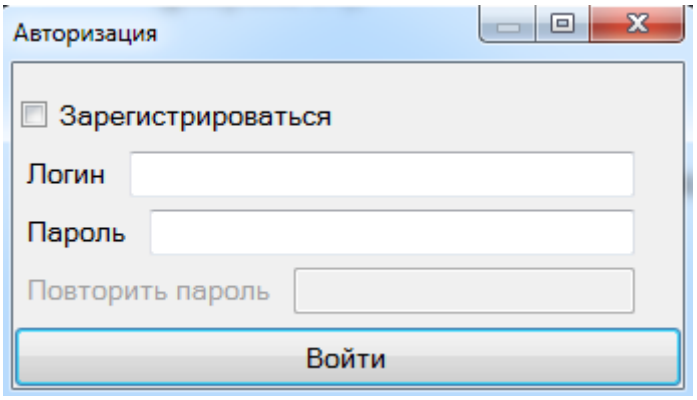

Рисунок 17 – Вид программы в момент запуска

При попытке войти, не вводя логин, напротив поля появляется сообщение об ошибке (рисунок 18).

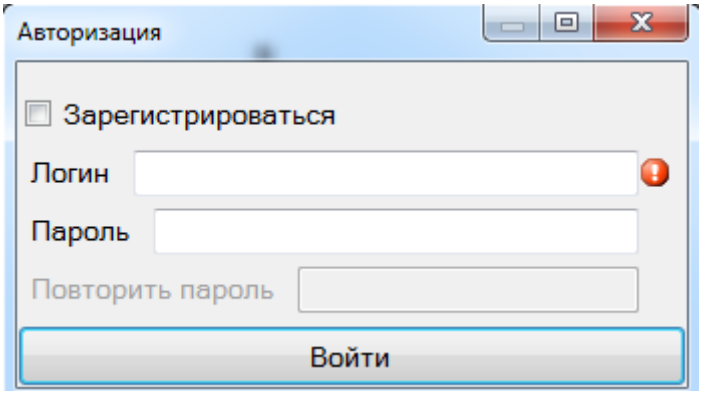

Рисунок 18 – Попытка войти без логина и пароля

Также сообщение об ошибке получаем, если попытаемся ввести логин в неверном формате, ввести логин без пароля, ввести неправильный логин или пароль. Если же логин и пароль введены правильно, окно авторизации закроется и откроется главная форма (рисунок 20).

Если включить флажок «Зарегистрироваться», станет доступным поле «Повторить пароль» и поменяется текст кнопки. Попробуем

зарегистрировать пользователя с логином Андрей, неправильно подтвердив пароль, получим сообщение об ошибке (рисунок 19).

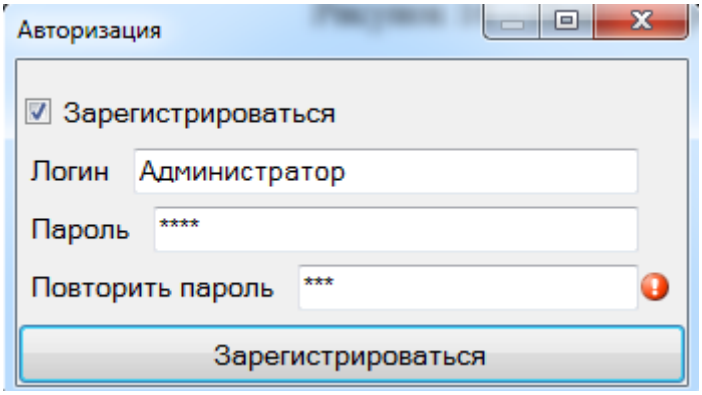

Рисунок 19 – Попытка регистрации без подтверждения пароля

Также ошибку получим, если введём логин в неверном формате, введём занятый логин, не введём логин или пароль. Если ввести логин и пароль правильно, а также подтвердить пароль, окно регистрации закроется и откроется главная форма (рисунок 20).

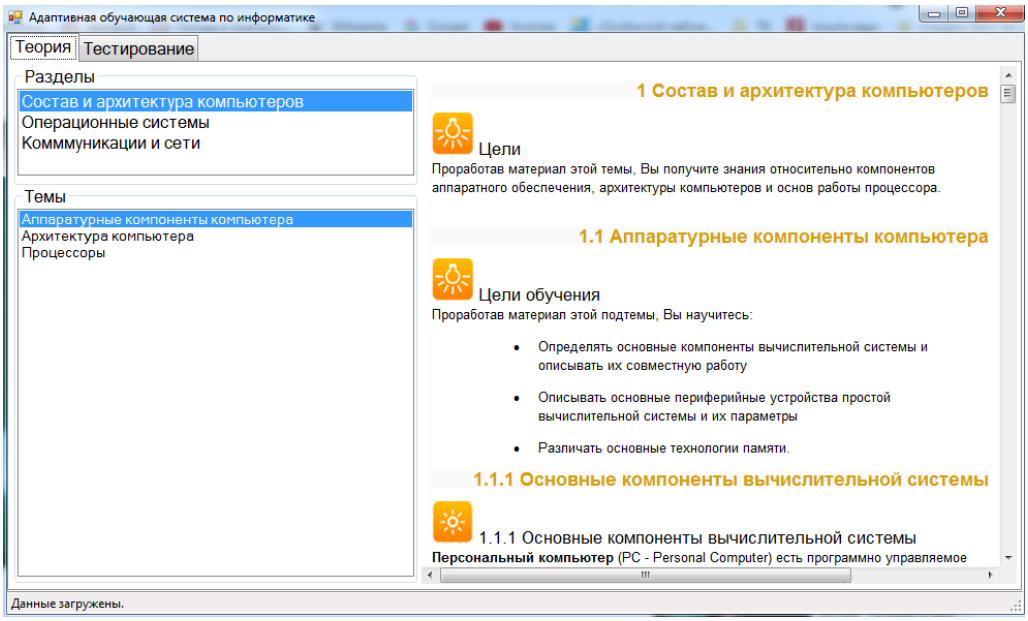

Рисунок 20 – Главная форма в момент открытия

Выберем в списке «Разделы» пункт «Операционные системы». В результате этого меняется содержимое списка «Темы», а в браузере открывается первая тема из этого списка – «Основные понятия» (рисунок 21).

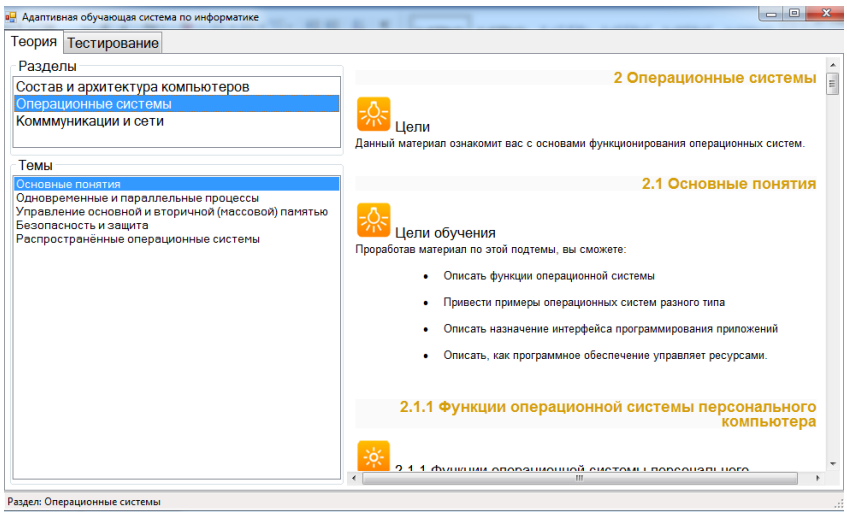

Рисунок 21 – Выбор раздела «Операционные системы»

Каждый раз, когда мы выбираем в списке «Темы» какой-то пункт, в браузере видим лекцию по этой теме. Когда мы выбираем пункт в списке «Разделы», в список «Темы» загружаются темы из выбранного раздела, а в браузер загружается первая тема в списке.

Переключимся на вкладку «Тестирование» (рисунок 22).

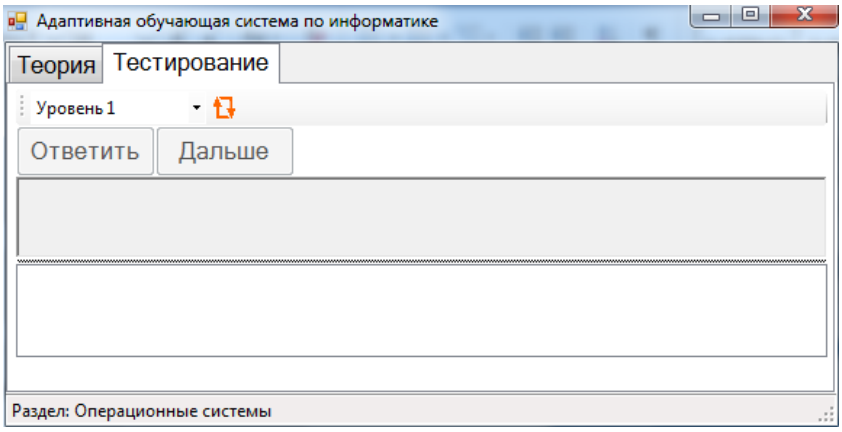

Рисунок 22 – Выбор вкладки «Тестирование»

Кликнем по кнопке возле выпадающего списка уровней. После этого кнопка «Ответить» становится доступной, в текстовой области появляется текст задания, а в списке – варианты ответа на него (рисунок 23).

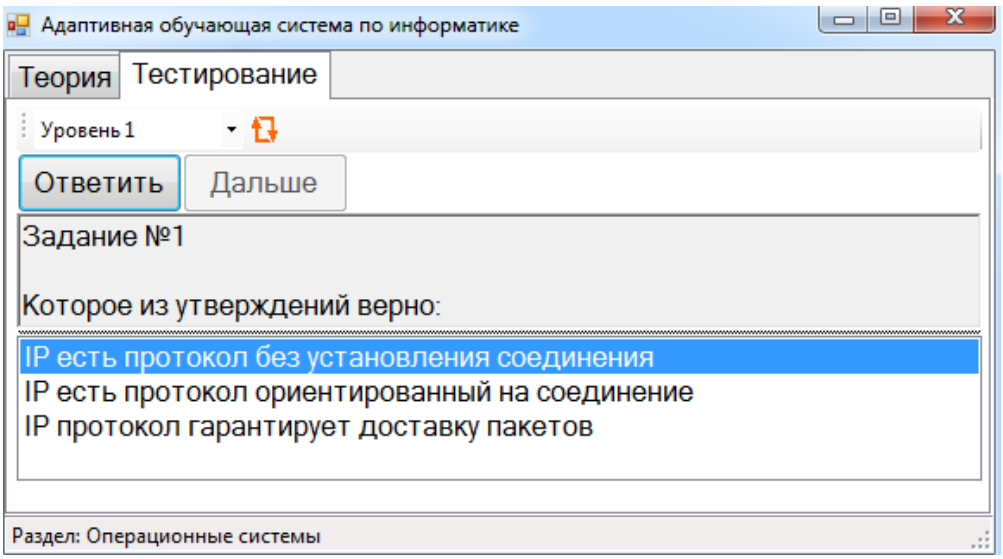

Рисунок 23 – Первое случайное задание теста

Кликнув по кнопке «Ответить», увидим сообщение о верном ответе (рисунок 24). Кнопка «Ответить» после этого станет недоступной, а кнопка «Дальше» – доступной.

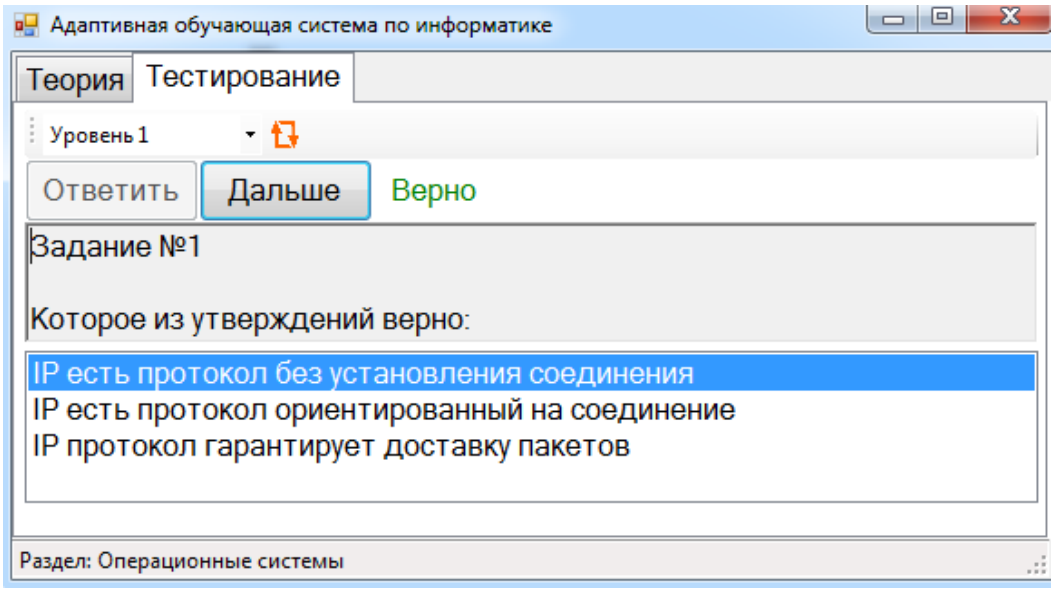

Рисунок 24 – Результат верного ответа

Кликнув по кнопке «Дальше», увидим второе случайное задание уровня 1. Кнопка «Дальше» снова отключается, а кнопка «Ответить» становится доступной. Если в каком-то задании мы сделаем ошибку, то получим соответствующее сообщение и увидим ссылку на лекцию по теме этого задания (рисунок 25).

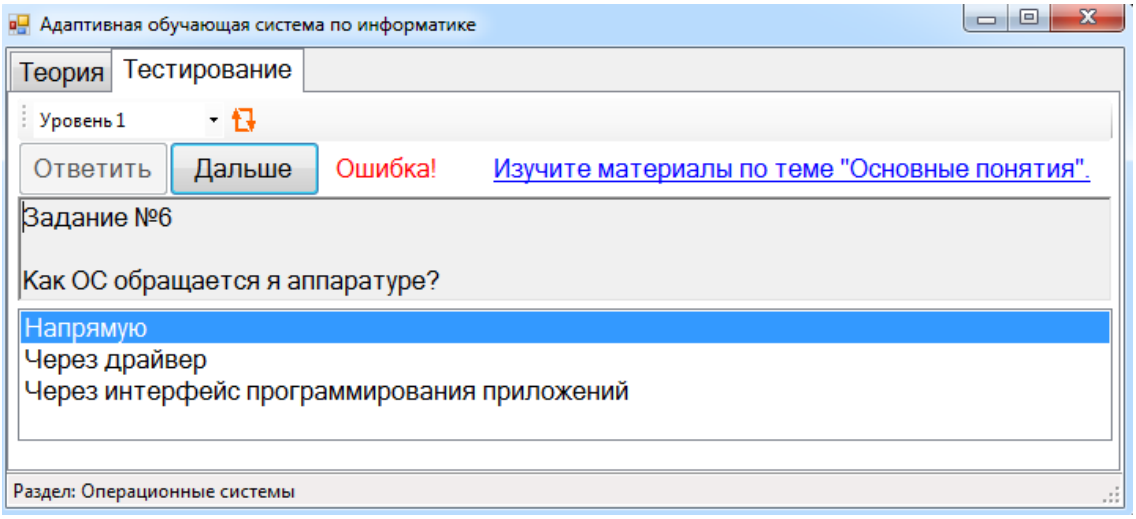

Рисунок 25 – Результат ошибочного ответа

Когда переходим по ссылке, откроется вкладка «Теория», очищается список тем, а в браузер загружается лекция по теме задания (рисунок 26).

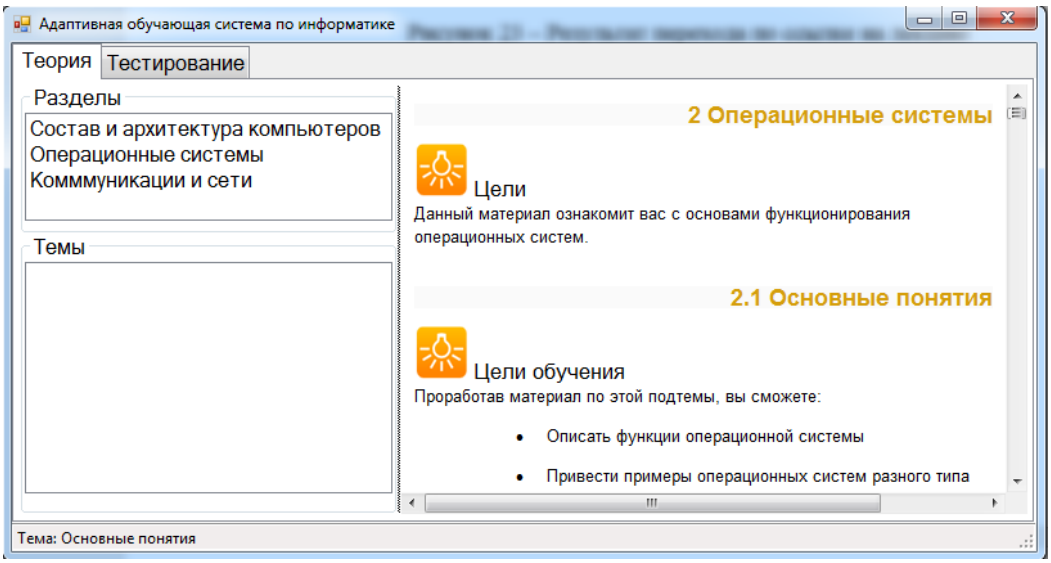

Рисунок 26 – Результат перехода по ссылке на лекцию

После ответа на последнее задание кнопки «Ответить» и «Дальше» становятся недоступным, список вариантов ответа очищается, а в текстовой области появляется результат тестирования (рисунок 27).

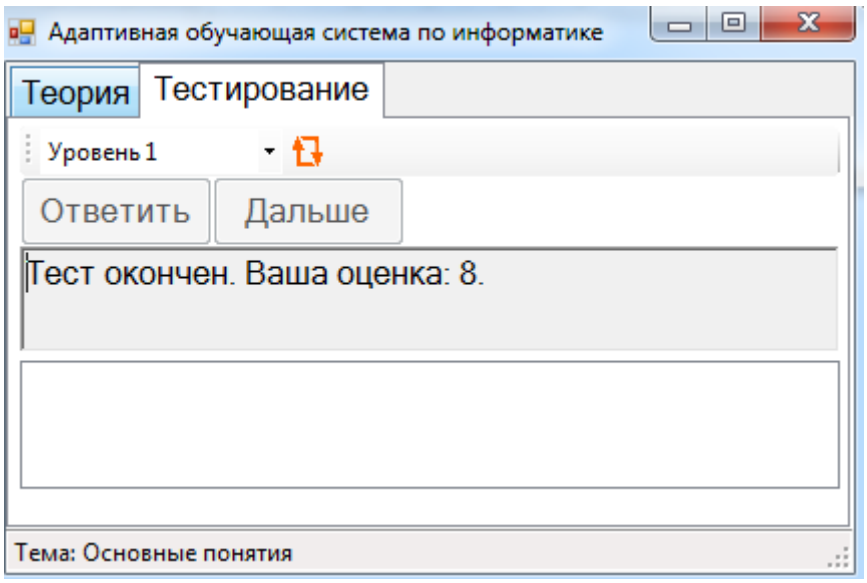

Рисунок 27 – Результат прохождения теста

Здесь мы продемонстрировали прохождения теста с заданием уровня 1. В тесте было 10 вопросов, но так как мы допустили две ошибки, то получили 8 баллов из 10 за этот тест. Если запустить тест, выбрав уровень 2, в тесте будет предложено 5 заданий по 2 балла за каждый правильный ответ.

В ходе испытаний программы ошибок в её работе обнаружено не было. Результаты её работы соответствуют ожиданиям.

3.2 Руководство пользователя системы

В момент запуска программы появляется окно авторизации (рисунок 17). Если у вас есть уже учётная запись в системе, в появившиеся поля вам следует ввести свои логин и пароль и нажать кнопку «Войти».

Если вы ещё не имеете учётной записи, включите флажок «Зарегистрироваться» для перехода в режим регистрации (рисунок 19). Придумайте логин и пароль. Логин – это непустая строка, состоящая из алфавитно-цифровых символов, между которыми допускается использовать пробелы. Зарегистрировать несколько пользователей с одинаковыми логинами невозможно. Пароль – это непустая строка из произвольного количества любых символов. В поле «Подтвердить пароль» следует ввести

ту же самую строку. Подтвердить регистрацию можно кнопкой «Зарегистрироваться».

Чтобы переключиться обратно в режим авторизации, следует отключить флажок «Зарегистрироваться».

После успешной авторизации или регистрации открывается главная форма (рисунок 20). На ней есть вкладки «Теория» и «Тестирование». Вкладка «Теории» по умолчанию активна. На вкладке «Теория» доступны лекции по разным темам. Для открытия нужной лекции следует выбрать соответствующие пункты в списках «Разделы» и «Темы». Размеры левой панели меню и правого окна просмотра лекций можно регулировать.

Прохождение тестов доступно на вкладке «Тестирование». Для начала теста выберите в выпадающем списке один из двух уровней сложности и нажмите кнопку напротив этого списка. Аналогичными тестами можно перезапустить тест в любой момент.

Для того, чтобы дать ответ на задание, следует выбрать в списке правильный вариант и нажать кнопку «Ответить». Если ответ будет ошибочным, появится ссылка, перейдя по которой, можно открыть лекцию по теме текущего задания. Для перехода к следующему заданию следует нажать кнопку «Дальше».

Оценка за тест станет доступна после его завершения.

3.3 Руководство системного программиста

FRUOPUS состоит из базы данных «tests.accdb», папки «Лекции» и приложения exe. Чтобы программа работала, эти файлы должны находиться в одном месте.

Базу данных можно открыть в приложении Microsoft Access. Она состоит из пяти таблиц, связи между которыми показаны на рисунке 10:

– Пользователи – рисунок 2;

– Разделы – рисунок 3;

– Темы – рисунок 4;

– Задания – рисунок 5;

– Варианты – рисунок 6.

Папка «Лекции» содержит файлы. Имя каждого файла состоит из номера темы и расширения htm.

Для запуска приложения требуется операционная система Windows 7 и новее; платформа .NET Framework 4 и новее.

Исходный код можно открыть в Visual Studio 2019 и новее. Он состоит из следующих файлов:

– Section.cs – класс для работы с разделами [приложение, 1];

– Theme.cs – класс для работы с темами [приложение, 2];

– Task.cs – класс для работы с заданиями [приложение, 3];

– Variant.cs – класс для работы с вариантами ответов [приложение, 4];

– User.cs – класс для работы с пользователями [приложение, 5];

– Test.cs – класс для работы с тестами [приложение, 6];

– InputException.cs – класс для работы с ошибками ввода в режиме авторизации или регистрации [приложение, 7];

– AuthForm.cs – описание обработчиков событий формы авторизации и регистрации [приложение, 8];

– AuthForm.Designer.cs – описание внешнего вида формы авторизации и регистрации;

– MainForm.cs – описание обработчиков событий главной формы [приложение, 9];

– MainForm.Designer.cs – описание внешнего вида главной формы;

– Program.cs – главный класс программы [приложение, 10].

3.4 Технико-экономическое обоснование программного продукта

Целью подведения технико-экономического обоснования приложения является распределение и обоснованность растраты ресурсов на разработку FruOpus. Данная разработка требует большого количества

материальных и трудовых затрат, а также небольшие затраты на расходные материалы. Затраты на разработку представлены в таблице 1 и таблице 2.

Таблица 1 Траты на разработку FruOpus.

№	Показатель	Единица измерения	Величина затрат
1	Затраты времени исполнителя на разработку	дней	98
2	Ставка дневной заработной платы исполнителя	Руб.	1000
3	Величина заработной платы исполнителя за выполнение	Руб.	98000
4	Ставка страхового взноса в Пенсионный фонд	%	7
5	Ставка страховых взносов в Фонд социального страхования	%	3
6	Тариф страховых взносов в Федеральный фонд обязательного медицинского страхования	%	5
7	Совокупный процент ставки страховых взносов	%	15
8	Величина страховых взносов	Руб.	5800
9	Страховые тарифы на обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний	%	0,3
10	Отчисления на обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний	Руб.	1000
11	Общая сумма страховых взносов	Руб.	6800
12	Суммарные затраты на оплату труда учетом страховых взносов	Руб.	104800

Таблица 2 Дополнительные траты на разработку FruOpus.

№	Наименование расходов	Единица измерения	Количество	Цена за единицу, руб.	Сумма затрат, руб.
1	Интернет	Месяц	7	840	5880
2	тетрадь	Шт.	4	95	380
3	Карандаш	Шт.	10	15	150
	Итого:				6410

Расчет полной себестоимости на разработку программного продукта представлен в таблице 3.

Таблица 3 Расчет полной себестоимости и цены договора на разработку программного продукта.

№	Наименование показателя	Единица измерения	Значение показателя
1	Заработная плата исполнителей с учетом страховых взносов	Руб.	104800
2	Дополнительные затраты	Руб.	6410
3	Полная себестоимость проекта	Руб.	111210
4	Средний уровень прибыльности (рентабельности) проектов разработки программных продуктов	%	30
5	Планируемый размер прибыли	Руб.	278025
6	Планируемая договорная цена разработки программного продукта	Руб.	58000
7	Фактическая цена разработки программного продукта	Руб.	58000

Выводы по главе 3

В данной главе было составлено и описано испытание системы, а также руководство пользователя и системного программиста. Были произведены расчеты и подсчеты для технико-экономического обоснования программного продукта (FruOpus), были составлены соответствующие таблицы.

ЗАКЛЮЧЕНИЕ

В процессе выполнения данной работы была разработана автоматизированная система для самостоятельной работы учащихся школы. Программа “FruOpus” реализована в виде приложения Windows Forms, работающей с базой данных Access. Были проведены испытания программы, в ходе которых не были выявлены ошибки. Составлено руководство пользователя и системного программиста по всем требованиям. Также были произведены расчеты технико-экономического обоснования проекта в ходе которого была посчитана сумма затрат на создание данного приложения.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Адаптивное обучение с моделью обучаемого / Л. А. Растригин, М. Х. Эренштейн. – Рига: Зинатне, 1988. – 160 с.: ил. ISBN 5-7966-0116-4.
2. Батан Л.Ф. Развитие познавательной активности в адаптивной технологии обучения: Курс лекций / Л.Ф. Батан. – Новосибирск: Изд-во НИПКиПРО, 2002. – 30 с. ISBN 5-87847-165-5.
3. Основы адаптивного обучения языку: Семиотич. аспекты развития речи с помощью автомата / Л. В. Шеншев. – Москва: Наука, 1995. – 111 с. ISBN 5-02-013036-2.
4. Система адаптивного обучения / Г. М. Анохина, Н. А. Морозова, С. А. Рогачев, Ю. А. Савинков; [Под ред. Ю. А. Савинкова]. – Воронеж: ВОИПКРО, 2000. – 167 с.: ил. ISBN 5-87938-110-2.
5. Современные образовательные технологии и методики: актуальные вопросы теории и практики / Центр образовательного и науч. консалтинга. – Чехов: Студия полиграфии, 2013. – 202 с.: ил. ISBN 978-5-905963-07-0.
6. Elitarium: Центр диагностического образования [сайт]. – 2023. – URL: http://www.elitarium.ru (дата обращения: 20.06.2023).
7. PikaTest [сайт]. – 2023. – URL: http://kripexx.narod.ru/pikatest (дата обращения: 20.06.2023).
8. UniTest [сайт]. – 2023. – URL: http://unitest.sfu-kras.ru/ (дата обращения: 20.06.2023).

ПРИЛОЖЕНИЯ

Приложение 1. Листинг файла Section.cs

```
using System.Collections.Generic;
namespace LB.Edu.Testing
{
    /// <summary>
    /// Тематический раздел.
    /// </summary>0
    public class Section
    {
        /// <summary>
        /// Список тем заданного раздела.
        /// </summary>
        private List<Theme> themes = null;

        /// <summary>
        /// Код раздела.
        /// </summary>
        public int ID { get; private set; }

        /// <summary>
        /// Название раздела.
        /// </summary>
        public string Name { get; private set; }

        /// <summary>
        /// Возвращает список тем заданного раздела.
        /// </summary>
        public List<Theme> Themes
        {
            get => themes is null? (themes = Theme.SelectAll(this)): themes;
        }

        /// <summary>
        /// Преобразует раздел в текст.
        /// </summary>
        /// <returns>Название раздела.</returns>
        public override string ToString() => Name;

        /// <summary>
        /// Отбирает все разделы из базы данных.
        /// </summary>
        /// <returns>Список отобранных разделов.</returns>
        public static List<Section> SelectAll()
        {
            Program.Cmd.CommandText = "SELECT * FROM [Разделы]";
            List<Section> result = new List<Section>();
            using (var reader = Program.Cmd.ExecuteReader())
                while(reader.Read())
                {
                    Section obj = new Section()
                    {
                        ID = reader.GetInt32(0),
                        Name = reader.GetString(1)
                    };
                    result.Add(obj);
                }
           return result;
        }
    }
```

```
}
```

Приложение 2. Листинг файла Theme.cs

```
using System.Collections.Generic;

namespace LB.Edu.Testing
{
    /// <summary>
    /// Тема.
    /// </summary>
    public class Theme
    {
        /// <summary>
        /// Код темы.
        /// </summary>
        public int ID { get; private set; }

        /// <summary>
        /// Название темы.
        /// </summary>
        public string Name { get; private set; }

        /// <summary>
        /// Раздел, к которому относится тема.
        /// </summary>
        public Section Section { get; private set; }

        /// <summary>
        /// Преобразует тему в текст.
        /// </summary>
        /// <returns>Название темы.</returns>
        public override string ToString() => Name;

        /// <summary>
        /// Отбирает тему с заданным кодом.
        /// </summary>
        /// <param name="id">Код темы.</param>
        /// <returns>Тема.</returns>
        public static Theme Select(int id)
        {
            Program.Cmd.CommandText = $"SELECT * FROM [Темы] WHERE [Код] = {id}";
            using (var reader = Program.Cmd.ExecuteReader())
                if (reader.Read())
                    return new Theme()
                    {
                        ID = reader.GetInt32(0),
                        Name = reader.GetString(1)
                    };
            return null;
        }

        /// <summary>
        /// Отбирает все темы заданного раздела.
        /// </summary>
        /// <param name="section">Раздел.</param>
        /// <returns>Список тем заданного раздела.</returns>
        public static List<Theme> SelectAll(Section section)
        {
            Program.Cmd.CommandText = $"SELECT * FROM [Темы] WHERE [Раздел] =
{section.ID}";
            List<Theme> result = new List<Theme>();
            using (var reader = Program.Cmd.ExecuteReader())
```

```
                while (reader.Read())
                {
                    Theme obj = new Theme()
                    {
                        ID = reader.GetInt32(0),
                        Name = reader.GetString(1),
                        Section = section
                    };
                    result.Add(obj);
                }
            return result;
        }
    }
}
```

Приложение 3. Листинг файла Task.cs

```
using System.Collections.Generic;

namespace LB.Edu.Testing
{
    /// <summary>
    /// Задание.
    /// </summary>
    public class Task
    {
        /// <summary>
        /// Код темы задания.
        /// </summary>
        private int themeID = 0;

        /// <summary>
        /// Тема задания.
        /// </summary>
        private Theme theme = null;

        /// <summary>
        /// Список вариантов ответа.
        /// </summary>
        private List<Variant> variants = null;

        /// <summary>
        /// Код задания.
        /// </summary>
        public int ID { get; private set; }

        /// <summary>
        /// Текст задания.
        /// </summary>
        public string Text { get; private set; }

        /// <summary>
        /// Уровень сложности.
        /// </summary>
        public byte Level { get; private set; }

        /// <summary>
        /// Индекс правильного ответа.
        /// </summary>
        public byte Answer { get; private set; }

        /// <summary>
        /// Возвращает список вариантов ответа.
        /// </summary>
        public List<Variant> Variants
        {
            get => variants is null ? (variants = Variant.Select(this)) : variants;
        }

        /// <summary>
        /// Возвращает тему задания.
        /// </summary>
        public Theme Theme
        {
            get => theme is null ? theme = Theme.Select(themeID) : theme;
        }

        /// <summary>
```

```
        /// Преобразует задание в текст.
        /// </summary>
        /// <returns>Текст задания.</returns>
        public override string ToString() => Text;

        /// <summary>
        /// Отбирает задания данного уровня.
        /// </summary>
        /// <param name="level">Номер уровня.</param>
        /// <returns>Список отобраных заданий.</returns>
        public static List<Task> Select(byte level)
        {
            Program.Cmd.CommandText = $"SELECT * FROM [Задания] WHERE [Уровень] =
{level}";
            List<Task> result = new List<Task>();
            using (var reader = Program.Cmd.ExecuteReader())
                while (reader.Read())
                {
                    Task obj = new Task()
                    {
                        ID = reader.GetInt32(0),
                        Text = reader.GetString(1),
                        themeID = reader.GetInt32(2),
                        Level = reader.GetByte(3),
                        Answer = reader.GetByte(4)
                    };
                    result.Add(obj);
                }
            return result;
        }
    }
}
```

Приложение 4. Листинг файла Variant.cs

```
using System.Collections.Generic;

namespace LB.Edu.Testing
{
    /// <summary>
    /// Вариант ответа.
    /// </summary>
    public class Variant
    {
        /// <summary>
        /// Код варианта ответа.
        /// </summary>
        public int ID { get; private set; }

        /// <summary>
        /// Текст варианта ответа.
        /// </summary>
        public string Text { get; private set; }

        /// <summary>
        /// Задание, к которому относится вариант ответа.
        /// </summary>
        public Task Task { get; private set; }

        /// <summary>
        /// Преобразует вариант ответа в текст.
        /// </summary>
        /// <returns>Текст варианта ответа.</returns>
        public override string ToString()
        {
            return Text;
        }

        /// <summary>
        /// Отбирает варианты ответа для данного задания.
        /// </summary>
        /// <param name="task">Задание.</param>
        /// <returns>Список вариантов ответа.</returns>
        public static List<Variant> Select(Task task)
        {
            Program.Cmd.CommandText = $"SELECT * FROM [Варианты] WHERE [Задание] =
{task.ID}";
            List<Variant> result = new List<Variant>();
            using (var reader = Program.Cmd.ExecuteReader())
                while (reader.Read())
                {
                    Variant obj = new Variant()
                    {
                        ID = reader.GetInt32(0),
                        Text = reader.GetString(1),
                        Task = task
                    };
                    result.Add(obj);
                }
            return result;
        }
    }
}
```

Приложение 5. Листинг файла User.cs

```
using System;
using System.Text.RegularExpressions;

namespace LB.Edu.Testing
{
    /// <summary>
    /// Пользователь.
    /// </summary>
    public class User
    {
        /// <summary>
        /// Логин.
        /// </summary>
        private string login = "";

        /// <summary>
        /// Пароль.
        /// </summary>
        private string password = "";

        /// <summary>
        /// Возвращает и задаёт код пользователя.
        /// </summary>
        public int ID { get; set; }

        /// <summary>
        /// Возвращает и задаёт логин пользователя.
        /// </summary>
        public string Login
        {
            get => login;
            set
            {
                login = Regex.Replace(value.Trim(), @"\s+", " ");
                if (login.Length == 0)
                    throw new Exception("Не задан логин!");
                if (!Regex.IsMatch(login, @"\w[\w ]+"))
                    throw new Exception("Логин имеет неверный формат!");
            }
        }

        /// <summary>
        /// Возвращает и задаёт пароль пользователя.
        /// </summary>
        public string Password
        {
            get => password;
            set
            {
                password = value;
                if (password.Length == 0)
                    throw new Exception("Не задан пароль!");
            }
        }

        /// <summary>
        /// Является ли пользователь администратором.
        /// </summary>
        public bool IsAdmin { get; private set; }

        /// <summary>
```

```
        /// Проверяет наличие пользователя в базе данных.
        /// </summary>
        /// <param name="withPass">Стоит ли учитывать пароль при проверке.</param>
        /// <returns>True - пользователь есть в базе данных;
        /// false - пользователя нет в базе данных.</returns>
        public bool Exists(bool withPass = true)
        {
            Program.Cmd.CommandText = "SELECT * FROM [Пользователи] "
               +  "WHERE [Логин] = @login ";
            var data = Program.Cmd.Parameters;
            data.Clear();
            data.AddWithValue("@login", login);

            if (withPass)
            {
                Program.Cmd.CommandText += "AND [Пароль] = @password";
                data.AddWithValue("@password", password);
            }

            using (var reader = Program.Cmd.ExecuteReader())
            if (reader.Read())
            {
                ID = reader.GetInt32(0);
                login = reader.GetString(1);
                password = reader.GetString(2);
                IsAdmin = reader.GetByte(3) == 1;
                return true;
            }
            return false;
        }

        /// <summary>
        /// Регистрирует пользователя в базе данных.
        /// </summary>
        public void Gegister()
        {
            Program.Cmd.CommandText = "INSERT INTO [Пользователи] " +
                "([Логин], [Пароль], [Тип]) VALUES(@login, @password, 0)";
            var data = Program.Cmd.Parameters;
            data.Clear();
            data.AddWithValue("@login", login);
            data.AddWithValue("@password", password);
            if (Program.Cmd.ExecuteNonQuery() == 1) Exists();
            else throw new Exception("Не удалось провести регистрацию!");
        }
    }
}
```

Приложение 6. Листинг файла Test.cs

```
using System;
using System.Collections.Generic;

namespace LB.Edu.Testing
{
    /// <summary>
    /// Тест.
    /// </summary>
    public class Test
    {
        /// <summary>
        /// Невыполненные задания теста.
        /// </summary>
        readonly List<Task> tasks = new List<Task>();

        /// <summary>
        /// Количество заданий.
        /// </summary>
        readonly byte count;

        /// <summary>
        /// Создаёт тест.
        /// </summary>
        /// <param name="level">Уровень сложности.</param>
        public Test(byte level)
        {
            LoasTasks(level, (byte)(10 / level));
            count = (byte)tasks.Count;
        }

        /// <summary>
        /// Возвращает текущее задание.
        /// </summary>
        public Task CurrentTask
        {
            get => tasks.Count > 0 ? tasks[0] : null;
        }

        /// <summary>
        /// Возвращает номер текущего задания.
        /// </summary>
        public int CurrentTaskNumber
        {
            get => count - tasks.Count + 1;
        }

        /// <summary>
        /// Загружает задания.
        /// </summary>
        /// <param name="level">Уровнь сложности.</param>
        /// <param name="count">Количество загружаемых заданий.</param>
        private void LoasTasks(byte level, byte count)
        {
            Random random = new Random();
            var all = Task.Select(level);
            for (byte i = 0; i < count; i++)
            {
                int index = random.Next(all.Count);
                Task task = all[index];
                tasks.Add(task);
                all.Remove(task);
```

```
            }
        }

        /// <summary>
        /// Сохраняет ответ на текущее задание.
        /// </summary>
        public void Answer()
        {
            tasks.RemoveAt(0);
        }
    }
}
```

Приложение 7. Листинг файла InputException.cs

```
using System;
using System.Windows.Forms;

namespace LB.Edu.Testing
{
    /// <summary>
    /// Ошибка ввода.
    /// </summary>
    public class InputException: Exception
    {
        /// <summary>
        /// Возвращает и задаёт элемент ввода.
        /// </summary>
        public Control Ctrl { get; private set; }

        /// <summary>
        /// Создаёт ошибку ввода.
        /// </summary>
        /// <param name="message">Сообщение об ошибке.</param>
        /// <param name="control">Элемент ввода, в котором произошла ошибка.</param>
        public InputException(string message, Control control):base(message)
        {
            Ctrl = control;
        }
    }
}
```

Приложение 8. Листинг файла AuthForm.cs

```
using System;
using System.Windows.Forms;

namespace LB.Edu.Testing
{
    /// <summary>
    /// Форма авторизции и регистрации.
    /// </summary>
    public partial class AuthForm : Form
    {
        /// <summary>
        /// Создаёт форму авторизации и регистрации.
        /// </summary>
        public AuthForm()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Выполняет регистрацию и авторизцию.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void BtnLogin_Click(object sender, EventArgs e)
        {
            try
            {
                Error.Clear();
                Program.User = new User();
                try { Program.User.Login = TextLogin.Text; }
                catch (Exception ex) {throw new InputException(ex.Message, TextLogin);
}
                try { Program.User.Password = TextPassword.Text; }
                catch (Exception ex) { throw new InputException(ex.Message,
TextPassword); }
                if (CheckReg.Checked)
                {
                    if (TextPassword.Text != TextConfirm.Text)
                        throw new InputException("Пароли не совпадают!", TextConfirm);
                    if(Program.User.Exists(false))
                        throw new InputException("Такой логин уже занят!", TextLogin);
                    Program.User.Gegister();
                    DialogResult = DialogResult.OK;
                }
                else if (Program.User.Exists()) DialogResult = DialogResult.OK;
                else throw new InputException("Неправильный логин или пароль!",
TextLogin);
            }
            catch (InputException ex)
            {
                Error.SetError(ex.Ctrl, ex.Message);
            }
            catch (Exception ex)
            {
                Error.SetError(TextLogin, ex.Message);
            }
        }

        /// <summary>
        /// Переключается между режимами регистрации и авторизации.
        /// </summary>
```

```
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void CheckReg_CheckedChanged(object sender, EventArgs e)
        {
            bool reg = CheckReg.Checked;
            label3.Enabled = TextConfirm.Enabled = reg;
            BtnLogin.Text = reg ? "&Зарегистрироваться" : "&Войти";
        }
    }
}
```

Приложение 9. Листинг файла MainForm.cs

```
using System;
using System.Drawing;
using System.IO;
using System.Windows.Forms;

namespace LB.Edu.Testing
{
    /// <summary>
    /// Главная форма.
    /// </summary>
    public partial class MainForm : Form
    {
        /// <summary>
        /// Тест.
        /// </summary>
        Test test;

        /// <summary>
        /// Оценка за тест.
        /// </summary>
        byte degree = 0;

        /// <summary>
        /// Создаёт главную форму.
        /// </summary>
        public MainForm()
        {
            AuthForm auth = new AuthForm();
            if (auth.ShowDialog() == DialogResult.OK)
            {
                InitializeComponent();
                try
                {
                    LabelResult.Text = Link.Text = "";
                    CmbLevel.SelectedIndex = 0;
                    ListSections.DataSource = Section.SelectAll();
                    Status.Text = "Данные загружены.";
                }
                catch(Exception ex)
                {
                    Status.Text = ex.Message;
                }
            }
            else Close();
        }

        /// <summary>
        /// Выводит текущее задание.
        /// </summary>
        private void ShowTask()
        {
            LabelResult.Text = Link.Text = "";
            Link.Tag = null;
            BtnNext.Enabled = false;
            AcceptButton = BtnAnswer;
            Task task = test.CurrentTask;
            if(task is null)
            {
                BtnAnswer.Enabled = false;
                RichTask.Text = $"Тест окончен. Ваша оценка: {degree}.";
                ListVariants.DataSource = null;
```

```
            }
            else
            {
                BtnAnswer.Enabled = true;
                RichTask.Text = $"Задание №{test.CurrentTaskNumber}\n\n{task}";
                ListVariants.DataSource = task.Variants;
            }
        }

        /// <summary>
        /// Выбирает тематический раздел.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void ListSections_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                Section section = ListSections.SelectedItem as Section;
                if (ListSections.SelectedItem is null) ListThemes.Items.Clear();
                else
                {
                    ListThemes.DataSource = section.Themes;
                    Status.Text = "Раздел: " + section;
                }
            }
            catch (Exception ex)
            {
                Status.Text = ex.Message;
            }
        }

        /// <summary>
        /// Выбирает тему лекции.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void ListThemes_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                Theme theme = ListThemes.SelectedItem as Theme;
                if (theme is null) return;
                string url =
$"{Directory.GetCurrentDirectory()}\\Лекции\\{theme.ID}.htm";
                Browser.Navigate(url);
                Status.Text = "Тема: " + theme;
            }
            catch (Exception ex)
            {
                Status.Text = ex.Message;
            }
        }

        /// <summary>
        /// Проверяет ответ пользователя.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void BtnAnswer_Click(object sender, EventArgs e)
        {
            try
            {
                Variant answer = ListVariants.SelectedItem as Variant;
```

```
                Task task = test.CurrentTask;
                test.Answer();
                BtnAnswer.Enabled = false;
                BtnNext.Enabled = true;
                AcceptButton = BtnNext;

                if (ListVariants.SelectedIndex + 1 == task.Answer)
                {
                    LabelResult.ForeColor = Color.Green;
                    LabelResult.Text = "Верно";
                    degree += task.Level;
                }
                else
                {
                    LabelResult.ForeColor = Color.Red;
                    LabelResult.Text = "Ошибка! ";
                    Link.Tag = task.Theme;
                    Link.Text = $"Изучите материалы по теме \"{task.Theme}\".";

                }
            }
            catch (Exception ex)
            {
                Status.Text = ex.Message;
            }
        }

        /// <summary>
        /// Показывает следующее задание.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void BtnNext_Click(object sender, EventArgs e)
        {
            try
            {
                ShowTask();
            }
            catch (Exception ex)
            {
                Status.Text = ex.Message;
            }
        }

        /// <summary>
        /// Открывает ссылку на теоретический материал.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Link_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            try
            {
                ListSections.SelectedIndex = -1;
                ListThemes.DataSource = null;
                Theme theme = Link.Tag as Theme;
                string url =
$"{Directory.GetCurrentDirectory()}\\Лекции\\{theme.ID}.htm";
                Browser.Navigate(url);
                Status.Text = "Тема: " + theme;
                Tabs.SelectedTab = tabPage1;
            }
            catch (Exception ex)
            {
```

```
                Status.Text = ex.Message;
            }
        }

        /// <summary>
        /// Перезапускает тест.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void ReloadTest(object sender, EventArgs e)
        {
            try
            {
                byte level = (byte)(CmbLevel.SelectedIndex + 1);
                if (level == 0) return;
                test = new Test(level);
                degree = 0;
                ShowTask();
            }
            catch (Exception ex)
            {
                Status.Text = ex.Message;
            }
        }
    }
}
```

Приложение 10. Листинг файла Program.cs

```
using System;
using System.Data.OleDb;
using System.Windows.Forms;

namespace LB.Edu.Testing
{
    /// <summary>
    /// Главный класс программы.
    /// </summary>
    static class Program
    {

        /// <summary>
        /// Строка подключения.
        /// </summary>
        static readonly string ConnectString =
            "Provider = Microsoft.ACE.OLEDB.12.0; Data Source=tests.accdb;";

        /// <summary>
        /// Подключение к базе данных.
        /// </summary>
        static OleDbConnection Connection;

        /// <summary>
        /// Запрос к базе данных.
        /// </summary>
        public static OleDbCommand Cmd;

        /// <summary>
        /// Возвращает и задаёт активного пользователя.
        /// </summary>
        public static User User { get; set; } = null;

        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            try
            {
                Connection = new OleDbConnection(ConnectString); //Создаём
подключение.
                Connection.Open(); //Открываем соединение с БД.
                Cmd = Connection.CreateCommand(); //Создаём запрос к БД.
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            try { Application.Run(new MainForm()); } catch { }
        }
    }
}
```