Министерство просвещения РФ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»

Институт математики, физики, информатики и технологий

Кафедра информатики, информационных технологий
и методики обучения информатике

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОВЕРКИ ЗНАНИЙ УЧАЩИХСЯ

Выпускная квалификационная работа
бакалавра по направлению подготовки
09.03.02 – Информационные системы и технологии

Исполнитель: студентка группы ИСиТ 1701z
Института математики, физики,
информатики и технологий
Гришина О.А.

Работа допущена к защите
«____»_____ 2022 г.
Зав. кафедрой _____

Руководитель: кандидат педагогических наук,
доцент кафедры ИИТ и МОИ
Рожина И.В

Екатеринбург – 2022

РЕФЕРАТ

Гришина О.А. РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОВЕРКИ ЗНАНИЙ УЧАЩИХСЯ, выпускная квалификационная работа: 72 стр., рис.9, табл. 2, библ.32 названия.

Ключевые слова: ПРОГРАММА, ШКОЛА, УЧАЩИЕСЯ, ПРОВЕРКА ЗНАНИЙ, ИНТЕРФЕЙС, ПРОГРАММИРОВАНИЕ НА C#, АВТОМАТИЗАЦИЯ.

Предмет разработки - приложение для автоматизации проверки знаний учащихся.

Цель разработки - сокращение времени педагогами МОУ СОШ №20 г. Каменска-Уральского на создание, проверку и оценку результата работ для текущего контроля знаний учащихся.

В данной работе описаны результаты создания программы для автоматизации проверки знаний учащихся.

Разработанная программа создавалась в среде разработки JetBrains Rider на языке программирования C#. Заполнение вопросов теста производится в самой программе.

В приложении создается тест, который распространяется на ПК учеников, далее при прохождении данного теста (выбор ответа из нескольких), вычисляется процентное соотношение правильных ответов к общему числу вопросов. Ответы сохраняются на ПК так же, как и выставленная оценка.

Оглавление

Введение

Мы живем в мире, где главенствует цифровизация. Сейчас сложно представить, что у кого-то нет мобильного телефона, компьютера, ноутбука или планшета – их отсутствие стало не мыслимым. Сложно игнорировать то, что у нас появился огромный ресурс для освобождения времени на более важные вещи. Не удивительно, что люди стараются упростить, автоматизировать большую часть процессов.

Сфера образования - это огромная ниша, куда необходимо внедрять информационные технологии для уменьшения нагрузки на работу учителей и преподавателей.

Период пандемии внес свои коррективы в процесс обучения: стандартный метод «взять листочки и писать самостоятельную работу» имеет ряд недостатков в оценке знаний учащихся, не удобен при проверки преподавателями и учителями, а также не исключает возможность списывания, именно поэтому у МОУ СОШ №20 г. Каменска-Уральского возникла потребность в разработке такого продукта, как приложение, автоматизирующее проверку знаний учащихся.

Предмет разработки является приложение для автоматизации проверки знаний учащихся.

Цель разработки - сокращение времени педагогами МОУ СОШ №20 г. Каменска-Уральского на создание, проверку и оценку результата работ для текущего контроля знаний учащихся.

Задачи разработки:

1. анализ и изучение сред разработки приложений;
2. изучение языка C#;
3. приложение;
4. внедрение приложения в МОУ СОШ №20 г. Каменска-Уральского.

Глава 1. Анализ предметной области

1.1 Педагогика. Ее основные задачи. Проблемы образовательного процесса для педагогов

Педагогика прошла громадный путь развития. Она превратилась в огромную разветвленную систему научных знаний пока проходила свой долгий, ветвистый путь. Так что же такое педагогика и какие проблемы у нее существуют? Рассмотрим такие определения как «педагог» и «педагогическая деятельность».

Педагог – это лицо, которое ведет практическую работу по воспитанию, обучению детей и(или) молодежи, а также имеющее специальную подготовку в этой области.

Педагогической деятельностью называют профессиональную деятельность, направленную на передачу социокультурного опыта путем обучения и воспитания.

Педагогическая деятельность как профессиональная имеет место в специально организованных обществом образовательных учреждениях: дошкольных заведениях, школах, профессионально-технических училищах, средних специальных и высших учебных заведениях, учреждениях дополнительного образования, повышения квалификации и переподготовки[4.].

Исходя из всего вышеперечисленного приходим к понимаю того, что объектом педагогики является образование, обучение как процесс.

Основные задачи педагогической деятельности:

- изучение и распространение педагогического опыта;
- прогнозирование дальнейшего развития образовательных систем;
- разработка новых методов, средств и форм обучения;
- внедрение результатов педагогических исследований в практику образования;
- обмен информацией с коллегами из других стран.

Основными проблемами школьного образовательного процесса можно назвать: отсутствие профильного обучения (интересы и склонности ученика не учитываются, задания по каждому предмету даются по максимуму, что снижает свободное время ребенка от процесса обучения и выполнения домашних заданий к минимуму); гонка за показателями (школы гонятся за рейтингом, именно поэтому на учеников идет давление по получению «хороших» оценок, в таких условиях чаще всего оценки и реальные знания не коррелируются); бюрократия (формализм пронизывает все сферы школьной жизни, это отрицать невозможно, детей с раннего возраста заставляют соблюдать огромное количество правил, порождая новые проблемы в области образования); разный уровень подготовки учеников (препятствовать появлению лидеров и отстающих невозможно в классической школьной системе, те, кто схватывает «налету» быстро начинают скучать, а вот аутсайдерам данного предмета будет нужен особый подход, который педагоги не всегда готовы обеспечить); низкая эффективность занятий (из сорока пятиминутного урока десять-пятнадцать минут это организационные вопросы, десять-пятнадцать минут конца занятия выделяется на проверку и разъяснение домашних заданий, получается что на подачу нового материала или закрепление прошлого уходит пятнадцать-двадцать минут); большие домашние задания (реальность такова, что те нормы, которые указаны в СанПине, сильно отличаются, поэтому ученики вынуждены жертвовать своим свободным временем, дабы вызубрить или написать кучу упражнений); устаревшие методы работы (возрастным педагогам сложнее приспособиться к электронному документообороту и программам)[8.].

Выделим основные проблемы в образовательном процессе для учителей и педагогов[6.]:

- отсутствие доверия к учителю, как к источнику информации (это связано с появлением интернета, общедоступностью любых материалов и обучающих программ);
- создание негативного образа учителя (ученики больше не видят авторитет в учителе, эту позицию формируют как СМИ, которые принижают роль учителя, также и родители, которые подают негативный пример своим поведением);
- учитель и ученики говорят на разных языках (большая часть учителей - преклонного возраста, игнорирующее существование и появление новых технологий, ученики же наоборот интересуются и используют их);
- трудности в воспитательной работе (следствие пункта 2);
- эмоциональное выгорание педагогов (у педагогов старше 35-40 лет пропадает энтузиазм в работе, нарастает негатив и усталость, это связанно с давлением общества, где каждый уверен и точно знает, что учитель обязан делать, а что ему категорически нельзя);
- огромная бюрократическая нагрузка (чаще всего заполнение различных отчетностей, всевозможных бумаг занимает колоссальное время педагога, которое может превышать в разы времени, тратящееся на взаимодействие с учениками).

1.2 Актуальность разрабатываемого проекта

В целях сокращения затраченного времени на создание, проверку, обработку проверочных работ учителями дирекцией МОУ СОШ № 20 г. Каменска-Уральского было принято решение о создании приложения для автоматизации проверки знаний учащихся. Приложение обязано снизить нагрузку на учителей, автоматизировать процесс оценки знаний учеников, а также моментально получать результат работы.

1.3 Постановка технического задания

1.3.1 Техническое задание

Техническое задание на разработку приложения для автоматизации проверки знаний учащихся.

Составлен на основе ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы»[9.].

1. Общие сведения.

1.1. Название организации-заказчика.

МОУ СОШ №20 г. Каменска-Уральского.

1.2. Название продукта разработки (проектирования).

«Система автоматизации проверки знаний учащихся»

1.3. Назначение продукта.

Приложение автоматизирует проверку знаний учащихся, сокращая затраты времени на составление тестов, проверку ответов большого количества учащихся, а также сводит к нулю возможность списывания.

1.4. Плановые сроки начала и окончания работ.

20.09.2021-31.05.2022.

2. Характеристика области применения продукта.

2.1. Процессы и структуры, в которых предполагается использование продукта разработки.

Данный проект предназначен для учителей МОУ СОШ №20 г. Каменска-Уральского.

2.2. Характеристика персонала (количество, квалификация, степень готовности).

В состав персонала необходимого для эксплуатации данного проекта необходимы:

- Учитель, составляющий тест;

• Лаборант, помогающий при возникновении проблем с использованием приложения, осуществляющий сбор информации;

• Ученики, использующие приложение.

Пользователи (учителя, ученики) должны уметь работать по инструкции. Лаборанту необходимо уметь маршрутизировать тесты и ответы, в соответствии с инструкцией.

3. Требования к продукту разработки.

3.1. Требования к продукту в целом.

Разрабатываемое приложение должно предоставлять ученикам тест, который создаст ранее учитель. Продукт будет анализировать ответы учащегося с правильными ответами, после нажатия клавиши «завершить тест» программа выведет сообщение с оценкой, а также сформирует документ с ответами ученика, процентным соотношением правильных ответов к вопросам и выставленной оценкой. В приложении обязательно должен использоваться герб школы. Приложение должно быть выполнено с красными оттенками в тон гербу, все надписи должны быть хорошо видны, читаемы.

3.2. Аппаратные требования.

• Персональный компьютер

• От 2 Gb оперативной памяти

• От 1 Gb свободного дискового пространства

3.3. Указание системного программного обеспечения (операционные системы, браузеры, программные платформы и т.п.).

• компоненты MicrosoftOffice

3.4. Указание программного обеспечения, используемого для реализации.

В данной работе использован язык программирования C#, а также среда разработки JetBrains Rider.

3.5. Форматы входных и выходных данных

На входе: Ввод вопросов с верными ответами.

На выходе: Оценка за самостоятельную учащихся, документ с ответами.

3.6. Источники данных и порядок их ввода в систему (программу), порядок вывода, хранения.

Пользователь-учитель создает в программе тест, при этом формируется два документа с форматами *_.crypt* (тест для учеников, открывается только в Examiner.exe) *u _.xml* (Созданный тест, который можно изменить, открывается только в ExamCreator.exe). При прохождении теста формируется документ с адресом «Мои документы/Examiner/Results.xlsx» с результатами ответов и оценкой.

3.7. Порядок взаимодействия с другими системами, возможности обмена информацией.

Не предусмотрено.

4. Требования к пользовательскому интерфейсу.

4.1. Общая характеристика пользовательского интерфейса.

Разрабатываемая модель должна быть реализована в виде готовой системы (приложения). Пользователь не может сам регулировать настройки, необходима поддержка разработчика.

4.2. Особенности ввода информации пользователем, представление выходных данных.

Тест составляется в приложении ExamCreator, где вносится сам вопрос, варианты ответа и выбирается правильный, либо несколько правильных вариантов. На выходе получаем документ с ответами ученика, а также сообщение с оценкой в самом приложении.

5. Требования к документированию.

5.1. Перечень сопроводительной документации.

Руководство пользователя-учителя «приложение для автоматизирования проверки знаний учащихся». Инструкция для ученика «приложение для автоматизации проверки знаний учащихся». Руководство для лаборанта «Приложение для автоматизации проверки знаний учащихся».

6. Порядок сдачи-приемки продукта.

В соответствии с планом выполнения ВКР (01.03.2021 – 01.02.2022).

1.3.2 Визуальное представление проекта

Схема работы приложения для создания тестов.

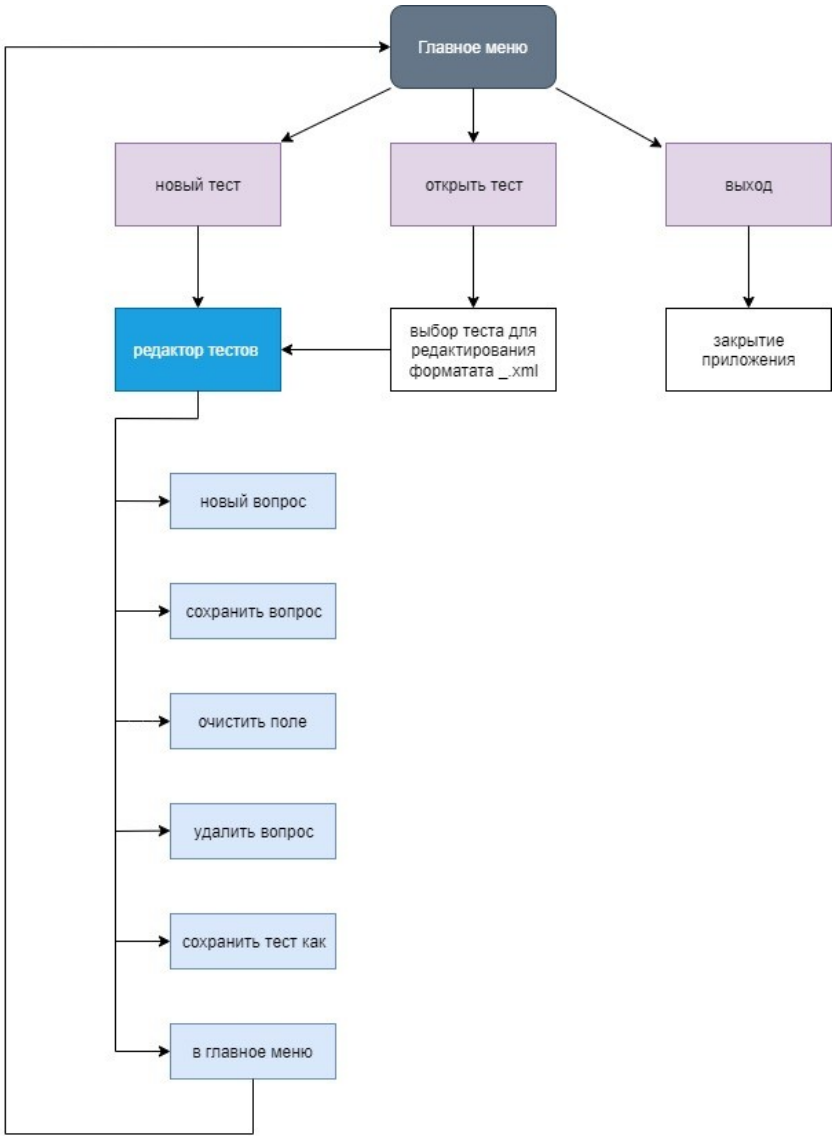
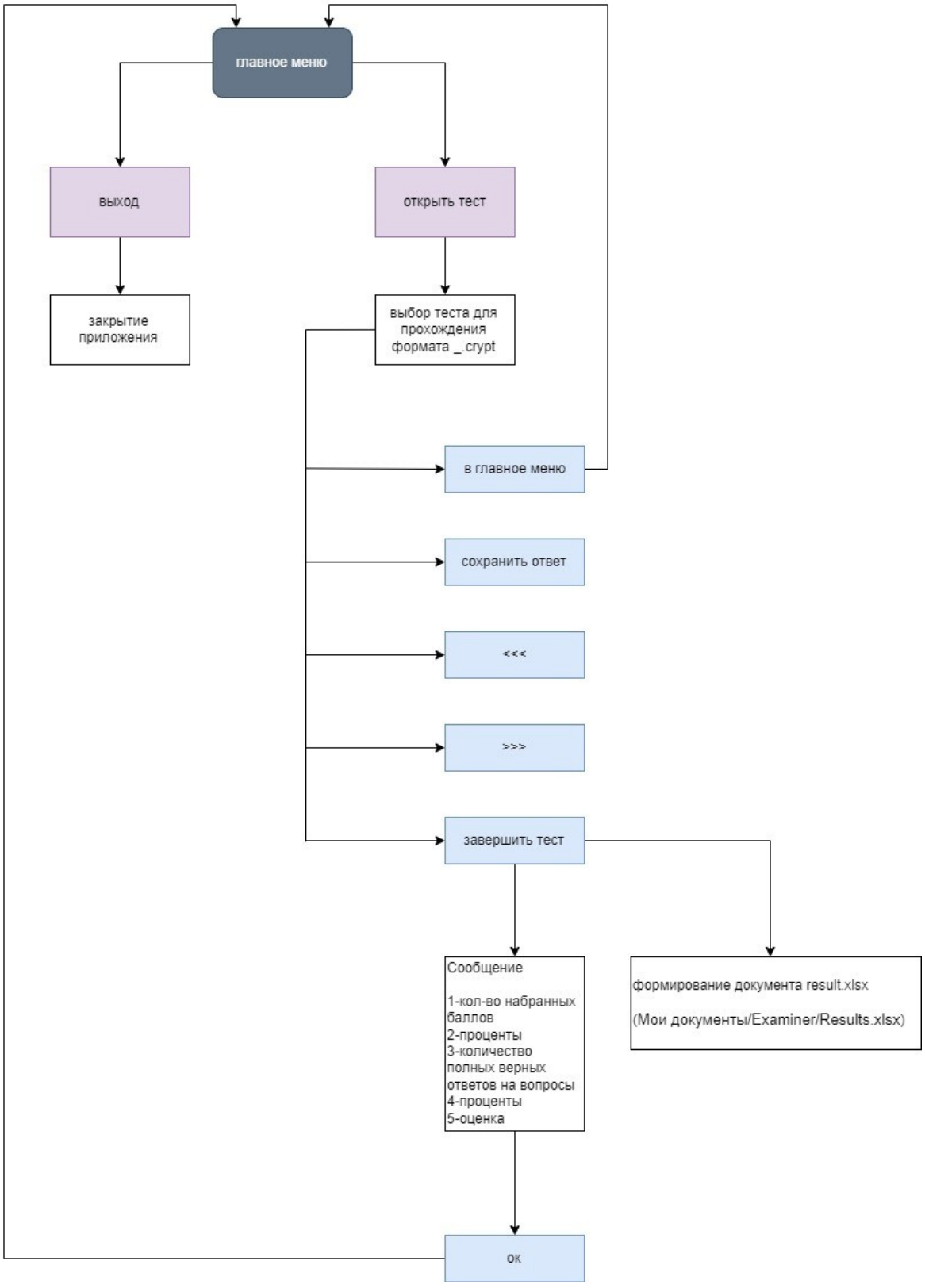


Схема приложения для прохождения тестов.



Разработка интерфейса приложений.

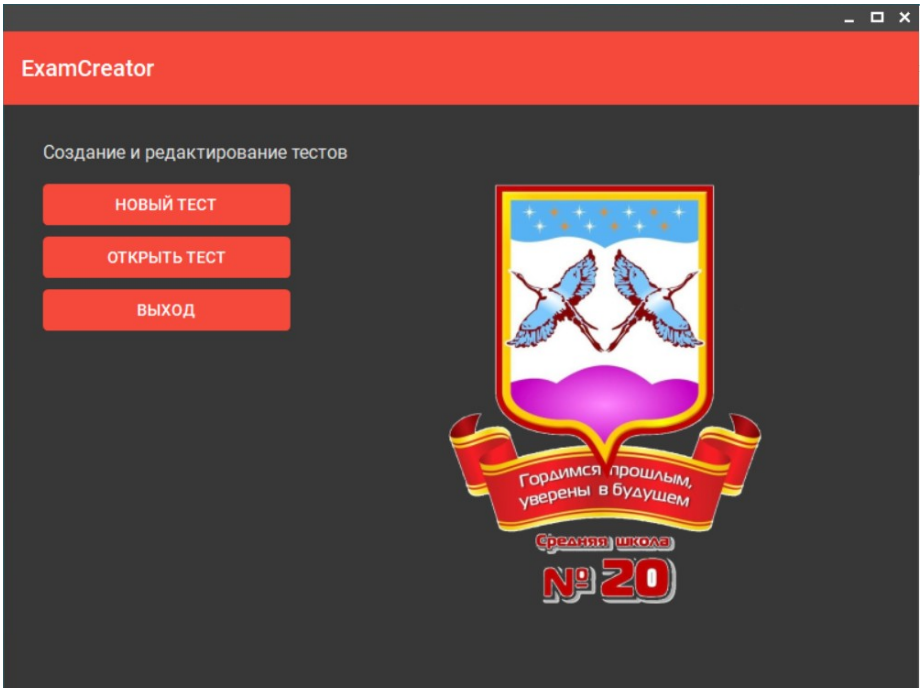


Рисунок 1: Интерфейс приложения для учителя - создания теста

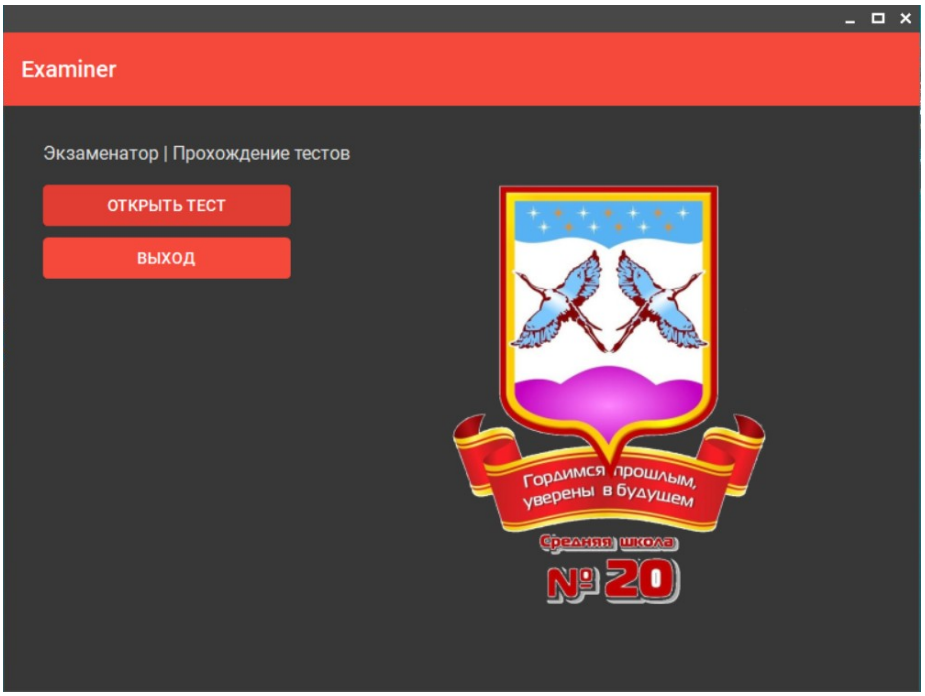


Рисунок 2: Интерфейс приложения для ученика - прохождение теста

Глава 2. Реализация проекта «Приложение для автоматизации проверки знаний учащихся»

2.1 Анализ технологий реализации

2.1.1 Выбор среды программирования

Для реализации проекта необходимо выбрать среду разработки и язык программирования.

Среда разработки Visual Studio .NET является интегрированной средой для создания, документирования, запуска и отладки программ, написанных на языках .NET. Visual Studio .NET – открытая языковая среда, в которую можно добавлять любые языки программирования, компиляторы которых создаются сторонними разработчиками (необходимым условием является использование Framework.NET)[12.].

Среда разработки JetBrains Rider – это быстрая, мощная кроссплатформенная IDE для .NET. Она позволяет разработать приложения .NET, ASP.NET, .NET Core, Xamarin и Unity на Windows, Mac и Linux. Rider поддерживает .NET Framework, новую платформу .NET Core и проекты на основе Mono. IDE позволяет разрабатывать десктопные приложения[10.].

Сравним эти две среды ниже:

Критерии сравнения	**Visual Studio**	**JetBrains Rider**
Работает для .NET и интерфейсного кода	+	+
Работает на Windows	+	+
Работает на macOC	-	+
Работает на Linux	-	+

Визуальное различие и слияние	+	+
Визуализация, сравнение, отмена прямо в редакторе	-	+
Возможность прерывания процессов слияния Git	-	+
Несколько репозиториев в одном проекте	-	+
Анатации кода для анализа нулевой способности	+	+
Индикация наследников типов и переопределений методов в редакторе	-	+

Сравнив две среды разработки выбираем JetBrains Rider, так как у него более расширенный функционал.

2.1.2 Выбор языка программирования

После того, как мы определились со средой программирования, необходимо определиться с языком программирования. Так как наша среда JetBrains Rider, то разработка приложения возможна или на C# или на C++.

Сравним эти два языка ниже в таблице.

Критерии сравнения	**C#**	**C++**
Быстрая скорость разработки	+	-
Большое количество	-	+

библиотек		
Удобство отладки кода	+	-
Простота синтаксиса	+	-
Удобство сборки	+	-

Исходя из сравнений в таблице выбираем C#, он проще и удобнее отлаживается, а так как сроки выполнения поставленных задач достаточно короткие, то скорость разработки является основополагающей.

C# - это C-подобный язык, который был создан в 1993- 2001 годах инженерами Microsoft (Андерс Хейлсберг и Скотт Вильтаумот – руководители проекта разработки языка) и многое взял от своих предшественников – Java, Basic, C++. Язык активно развивается, в него постоянно добавляются синтаксические конструкции, что заметно упрощает работу программиста, а также увеличивает скорость разработки и делает код удобно-читаемым. Его все время совершенствуют, увеличивая надежность и быстродействие, а так же вводят много нового (лямбды, асинхронные методы, динамическое связывание). Новшеством платформы .NET была технология активных серверных страниц, с помощью которой быстро разрабатывались вэб-приложения[14.].

C# уже давно поддерживает много полезных функций:

- полиморфизм,
- инкапсуляция,
- наследование,
- перезагрузка операторов,
- статическая типизация.

2.2 Разработка приложения

Нами разработано приложение для автоматизации проверки знаний учащихся для учителей МОУ СОШ №20 г. Каменска-Уральского. Приложение

фактически состоит из двух взаимосвязанных приложений: ExamCreator.exe и Examiner.exe, где первый предназначен для создания тестов преподавателями, а второй – для прохождения тестов учениками.

Приложение ExamCreator.exe.

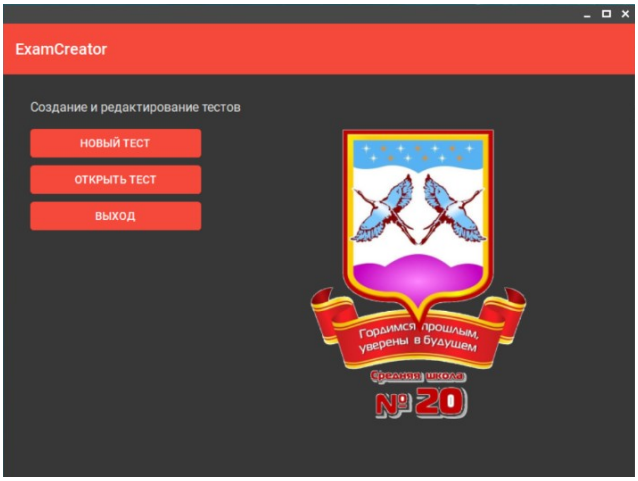


Рисунок 3: Приложение "ExamCreator"

Для начала работы необходимо открыть файл ExamCreator.exe. В главном меню находятся следующие кнопки: «новый тест», «открыть тест» и «выход».

При нажатии на «новый тест» открывается редактор теста. Он состоит из поля, для введения вопроса, четырех полей с маркерами для введения ответов и выборки верного/верных, а также кнопок «новый вопрос», «сохранить вопрос», «очистить поле», «удалить вопрос», «сохранить тест как» и «главное меню».

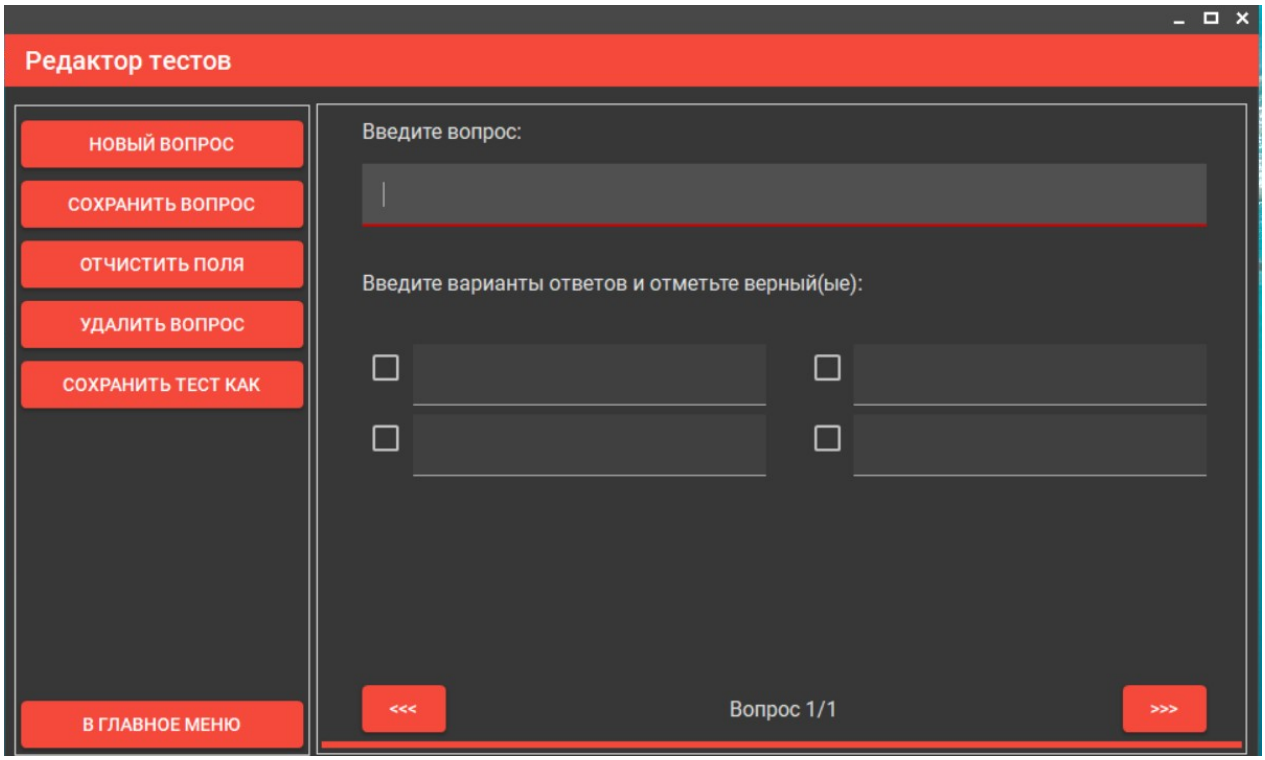


Рисунок 4: Редактор тестов в "ExamCreator"

Рассмотрим план создания теста, например, из трех вопросов под названием «Числа»:

- Открыть ExamCreator.exe;
- Нажать «новый тест»;
- Нажать «новый вопрос»;
- Вводим в поле вопрос;
- Вводим в поля для ответов ответы от 2 до 4;
- Выбираем правильные ответы;
- Нажимаем «сохранить вопрос»;
- Чтобы ввести второй вопрос выполняем пункты 3-7;
- Для ввода третьего вопроса выполняем пункты 3-7;
- Нажимаем «сохранить как»;
- Вводим название теста «Числа»;
- На рабочем столе создаются два файла с названиями *Числа.xml* и *Числа.crypt*;

- Создание теста с названием «Числа» окончено.

При нажатии на «очистить поле» очистятся все поля данного вопроса, это может понадобиться при редактировании тестов. Если же из теста необходимо будет удалить какой-либо вопрос, то необходимо нажать кнопку «удалить вопрос».

Тело программы.

```
using System;
using MaterialSkin;
using MaterialSkin.Controls;

namespace ExamCreator.Forms
{
    /// <summary>
    /// Главная форма (Главное меню)
    /// </summary>
    public partial class MainForm : MaterialForm
    {
        /// <summary>
        /// Переменная для отслеживания открытия теста
        /// </summary>
        private bool _opening;

        /// <summary>
        /// Конструктор главной формы
        /// </summary>
        public MainForm()
        {
            InitializeComponent();

            // Настройка цветовой темы для окон
            var materialSkinManager =
MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);
```

```
                materialSkinManager.Theme =
MaterialSkinManager.Themes.DARK;
                materialSkinManager.ColorScheme = new
ColorScheme(
                    Primary.Red500,
                    Primary.Grey800,
                    Primary.Red900,
                    Accent.Red700,
                    TextShade.WHITE);
            }

            /// <summary>
            /// Нажатие на кнопку "Новый тест"
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
            private void bNew_Click(object sender, EventArgs e)
            {
                // Создаем редактор для нового теста
                _opening = false;
                var editor = new Editor(ref _opening);

                // Отображаем едактор, скрываем меню
                editor.Show();
                Hide();
            }

            /// <summary>
            /// Нажатие на кнопку "Открыть тест"
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
            private void bOpen_Click(object sender, EventArgs e)
            {
```

```
                // Создаем редактор для открытого теста
                _opening = true;
                var editor = new Editor(ref _opening);

                // Если пользователь прервал открытие, то выходим
из функции
                if (!_opening)
                {
                    return;
                }

                // Отображаем редактор, скрываем меню
                editor.Show();
                Hide();
            }

            /// <summary>
            /// Нажатие на кнопку "Выход"
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
            private void bExit_Click(object sender, EventArgs e)
            {
                // Закрыть MainForm = закрыть все
                Close();
            }
        }
    }
```

Приложение Examiner.exe.

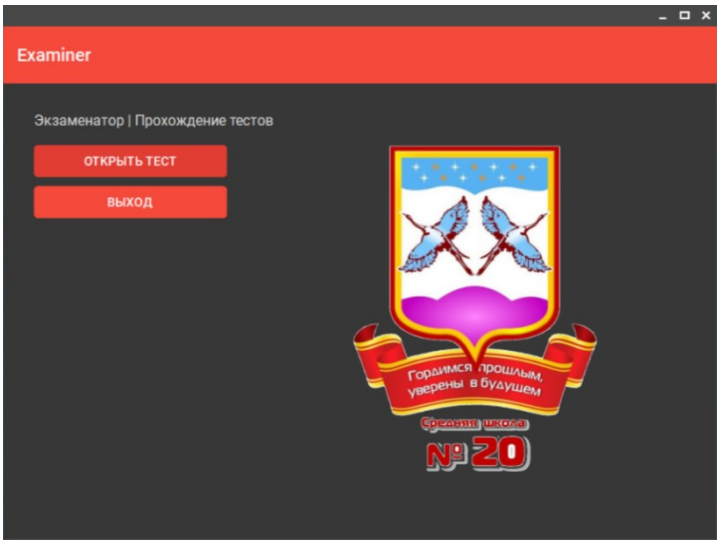


Рисунок 5: Приложение "Examiner"

Для начала работы необходимо открыть файл Examiner.exe. В главном меню находятся кнопки «открыть тест» и «выход». После нажатия на «открыть тест» откроется окно, где нужно выбрать то название теста, которое заранее указал преподаватель, после чего файл прогрузится в приложение и можно будет переходить к его выполнению. В приложении есть кнопки «в главное меню», «сохранить ответ», «<<<», «>>>» и «завершит тест». Для перехода к следующему или предыдущему вопросу используются кнопки «>>>» и «<<<» соответственно. Выбор ответа происходит путем нажатия маркера, после чего обязательно нажать кнопку «сохранить ответ», если же этого не сделать, то при переходе к следующему вопросу поля будут очищены, а ответ будет защитан как неверный. Тест считается завершенным если на все вопросы пользователь-ученик дал ответ и убедился в том, что поля заполнены, а также нажал кнопку «завершить тест», при этом программа выдаст сообщение с указанием количества набранных баллов и их процентов, количество полных верных ответов и их процентов, а также оценку за работу. В момент завершения теста программа сформирует документ с результатами, который будет иметь путь c://Мои документы/Examiner/Results.xlsx.

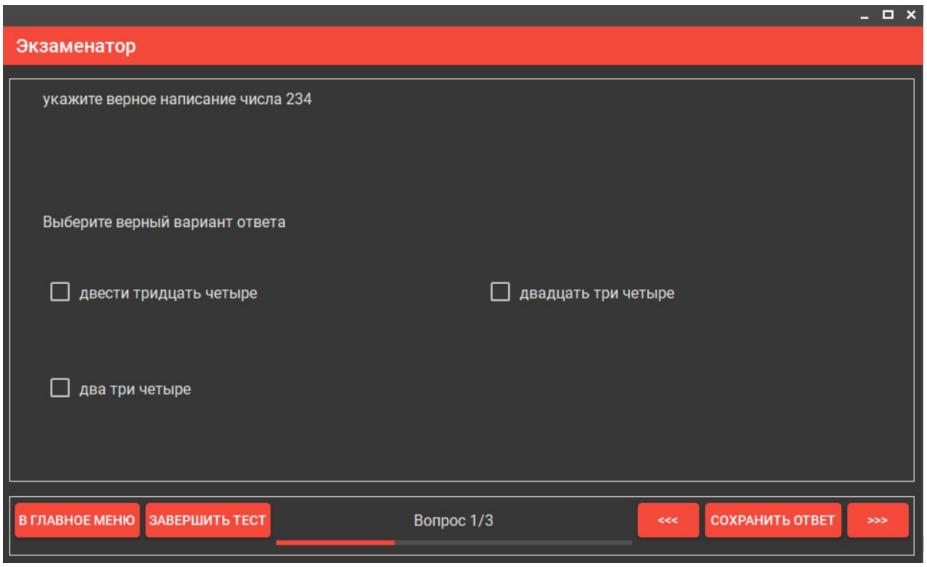


Рисунок 6: Интерфейс приложения "Examiner"

Чтобы пройти тест, например, «Числа», нужно нажать на «открыть тест», далее выбрать файл *Числа.crypt*. Будем считать, что ответы давались сразу, тогда после выбора ответов была нажата кнопка «сохранить ответ», а после третьего вопроса – «сохранить ответ» и «завершить тест». Программа выдаст сообщение с указанием количества набранных баллов и их процентов, количество полных верных ответов и их процентов, а также оценкой за работу. Тестирование завершено.

Тело программы.

```
using System;
using MaterialSkin;
using MaterialSkin.Controls;

namespace Examiner.Forms
{
    /// <summary>
    /// Главная форма (Главное меню)
    /// </summary>
    public partial class MainForm : MaterialForm
    {
        /// <summary>
```

```
        /// Переменная для отслеживания открытия теста
        /// </summary>
        private bool _opening;

        /// <summary>
        /// Конструктор главной формы
        /// </summary>
        public MainForm()
        {
            InitializeComponent();

            // Настройка цветовой темы для окон
                                    var   materialSkinManager   =
MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);
                                    materialSkinManager.Theme   =
MaterialSkinManager.Themes.DARK;
            materialSkinManager.ColorScheme = new ColorScheme(
                Primary.Red500,
                Primary.Grey800,
                Primary.Red900,
                Accent.Red700,
                TextShade.WHITE);
        }

        /// <summary>
        /// Нажатие на кнопку "Открыть тест"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bOpen_Click(object sender, EventArgs e)
        {
            // Создаем экзаменатор для открытого теста
            _opening = true;
```

```
            var tester = new Tester(ref _opening);

              // Если пользователь прервал открытие, то выходим из
функции
            if (!_opening)
            {
                return;
            }

            // Отображаем экзаменатор, скрываем меню
            tester.Show();
            Hide();
        }

        /// <summary>
        /// Нажатие на кнопку "Выход"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bExit_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

2.3 Сопроводительная документация

2.3.1 Руководство пользователя-учителя «Приложение для автоматизирования проверки знаний учащихся»

1. Запуск программы.

Для запуска программы необходимо открыть приложение ExamCreator.exe, расположенное на Вашем рабочем столе, если его нет, то обратитесь к лаборанту.

2. Интерфейс. Работа с интерфейсом.

В главном меню находятся следующие кнопки: «новый тест», «открыть тест» и «выход». При нажатии на «новый тест» открывается редактор теста. Он состоит из поля, для введения вопроса, четырех полей с маркерами для введения ответов и выборки верного/верных, а также кнопок «новый вопрос», «сохранить вопрос», «очистить поле», «удалить вопрос», «сохранить тест как» и «главное меню».

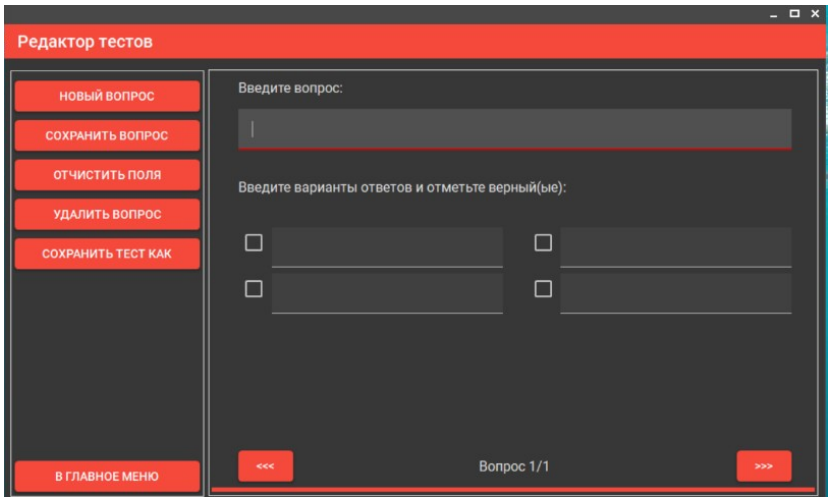


Рисунок 7: Редактор тестов

Создание теста:

1) Выбрать «новый тест»;
2) Выбрать «новый вопрос»;
3) Ввод вопроса;
4) Ввод вариантов ответа;
5) Отметить верные варианты ответа;
6) Выбрать «сохранить вопрос»;
7) Повтор пунктов 2-6 до тех пор, пока не создастся необходимый тест;
8) Выбрать «сохранить тест как» и написать название Вашего теста.

Создание теста окончено.

При необходимости редактирования уже созданного теста:

1) Выбрать «главное меню»;
2) Выбрать «открыть тест» и нажать на тот тест, что будете редактировать;
3) Кнопками «<<<», «>>>» найдите вопрос, в который будете вносить изменения;
4) Если Вам необходимо удалить вопрос, то нажимаете «удалить вопрос», если необходимо изменить сам вопрос, то нажатие кнопки «очистить поле» удалит все записи из формы, после чего можете заполнить пустые поля (смотрите пункты 3-6 в «Создание теста»);
5) Ввод «сохранить тест как», переименовать тест, например, дописать в названии цифру 1.

Редактирование теста окончено.

Перед самостоятельной работой необходимо связаться с лаборантом для распространения теста на ПК учеников.

2.3.2 Инструкция для учеников «Приложение для автоматизации проверки знаний учащихся»

Уважаемые ученики, для успешного прохождения теста просим Вас ознакомиться с основным функционалом приложения Examiner.

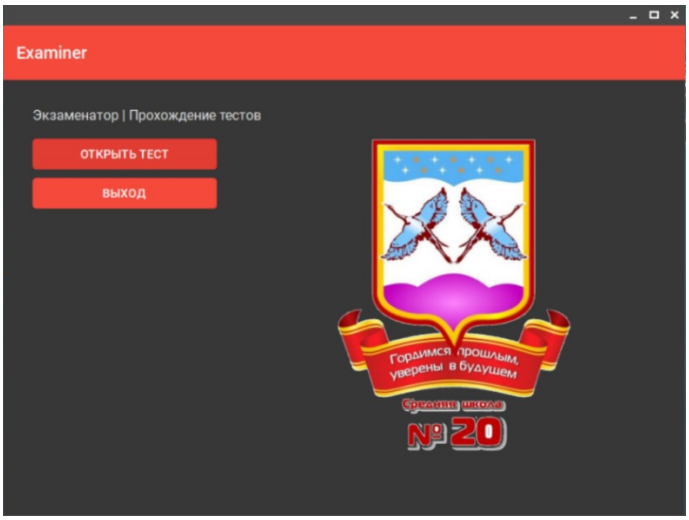


Рисунок 8: Интерфейс приложения "Examiner"

Для запуска нажмите Examiner.exe на рабочем столе. Если такого приложения нет, то сообщите об этом лаборанту или учителю. Чтобы начать тест нажмите кнопку «открыть тест», далее выберите тест с тем названием, который ранее сообщил Вам учитель.

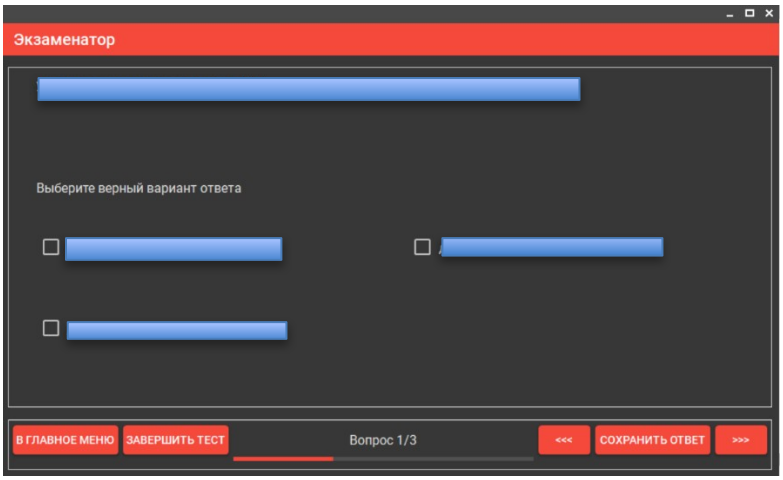


Рисунок 9: Интерфейс приложения, прохождене теста

Кнопки и их значения при прохождении теста:

- «в главное меню» - возвращает на главную страницу приложения;
- «завершить тест» - завершает тестирование, не выдает ошибок, если нет ответов на какие-либо вопросы;

- «<<<» - переход к предыдущему вопросу;
- «>>>» - переход к следующему вопросу;
- «сохранить ответ» - сохраняет ответ, при переходах к вопросам предыдущим или последующим не сбрасывает выставленные галочки, учитывается при завершении теста.

Чтобы пройти тест, нужно нажать на «открыть тест», далее выбрать файл _.*crypt*. После выбора ответов, которые считаете верными (нажмите на квадратик напротив него/них), нажимается «сохранить ответ». Для завершения теста и получения результатов выберите «завершить тест». Программа выдаст сообщение с указанием количества набранных баллов и их процентов, количество полных верных ответов и их процентов, а также оценкой за работу.

Внимание!!! Не нажимайте «ок» до тех пор, пока информацию не получит учитель или лаборант, а также просим проверить весь тест перед нажатием «завершения теста», чтобы избежать ошибок в заполнении теста и выставления неверной оценки Вашим знаниям.

Тестирование завершено! Успехов в учебе!

2.3.3 Руководство для лаборанта «Приложение для автоматизации проверки знаний учащихся».

1. Требования к аппаратному обеспечению.

Убедитесь, что все компьютеры соответствуют аппаратным и программным требованиям:

- 2 Gb оперативной памяти и более,
- 1 Gb свободного дискового пространства и более,
- компоненты MicrosoftOffice.

2. Сопровождение приложения

Установите пакет «ExamCreator» на компьютеры учителей, создайте ярлык на рабочем столе приложения ExamCreator.exe.

Установите пакет «Examiner» на компьютеры учеников, создайте ярлык на рабочем столе приложения Examiner.exe.

Ознакомьтесь с руководством пользователя-учителя «Приложение для автоматизирования проверки знаний учащихся», а также с инструкцией для учеников, чтобы в случае необходимости помочь учителям в создании теста или ученикам для верного ввода информации в поля.

Перед проведением теста файл с расширением _.*crypt*, находящийся на рабочем столе учителя распространите на ПК учеников.

После завершения теста переименуйте документы Results.xlsx с путем C://Мои документы/Examiner/Results.xlsx на фамилии учеников, соберите их в одну папку, укажите номер класса, фамилию учителя и дату теста, после чего передайте преподавателю.

Заключение

Для разработки приложения для автоматизации проверки знаний учащихся была использована среда разработки JetBrains Rider и язык программирования C#.

Это был заказ от директора МОУ СОШ №20 г. Каменска-Уральского, этим диктуется актуальность проекта.

Перед началом разработки была составлена схема разрабатываемого продукта, написано техническое задание, разработан интерфейс программы.

В результате написания выпускной квалификационной работы было разработано приложение для автоматизации проверки знаний учащихся. Продукт был выполнен в соответствии с техническим заданием. Заказчик удовлетворен работой, что подтверждает акт о внедрении.

Список информационных источников

1. Microsoft Corporation. Принципы проектирования и разработки программного обеспечения. Учеб. курс MCSD. М.: Изд.-торг. дом «Русская редакция», 2000.
2. Pros and Cons of Using C# as Your Backend Programming Language //Agilites:Software Development Company URL: https://agilites.com/pros-and-cons-of-using-c-as-your-backend-programming-language.html/(дата обращения: 02.10.2021).
3. Агуров П. В. C#. Разработка компонентов в MC Visual studio 2005/2008 / Агуров П. В. – СПб.:БХВ-Петербург, 2008. – 479 с.
4. Алдакимов, А.Н. Тенденции развития современной системы образования в России: историко – педагогический и социальный аспекты / А.Н. Алдакимов // Проблемы современного педагогического образования. 2017. № 57-3. С. 3-9.
5. Березин Б.И., Березин С.Б. Начальный курс С и С++. — М.: ДИАЛОГ-МИФИ, 1996.
6. Боровкова Т.И., Морев И.А. Мониторинг развития системы образования. Часть 2. Практические аспекты: Учебное пособие. - Владивосток: Изд-во Дальневосточного университета, 2004. - 134с.
7. Гагарина Л.Г. Технология разработки программного обеспечения: учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул. – М.: ИД «ФОРУМ»: ИНФРА-М, 2009.
8. Галактионова, Ю.Ю. Состояние системы образования в современной России и прогнозирование ее дальнейшего развития / Ю.Ю. Галактинова // Аллея науки. 2018. Т. 4. № 1 (17). С. 795-797.
9. ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению». – М.: Издательство стандартов, 1989.

10. Знакомство с Rider [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/ru-ru/rider/documentation/ (дата обращения: 20.09.2021).
11. Ишкова Э. А. Самоучитель C#. Начала программирования [Текст]: учебное пособие / Э. А. Ишкова — 2-е изд. — Санкт-Петербург: Наука и Техника, 2013. — 496 с.
12. Кариев Ч.А. Разработка Windows-приложений на основе Visual C# [Электронный ресурс]: учебное пособие / Ч.А. Кариев. – Электрон. текстовые данные. – Москва, Саратов: Интернет-университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. -768 с.
13. Катаев М.Ю. Объектно-ориентированное программирование: Учебное пособие. – Томск: Томский межвузовский центр дистанционного образования, 2000. – 145 с.
14. Котов О.М. Язык C#. Краткое описание и введение в технологии программирования [Текст]: учебное пособие / О.М. Котов – Екатеринбург: Урал. ун-та, 2014. – 208 с.
15. Крэг Ларман. Применение UML и шаблонов проектирования. – М. Издательский дом «Вильямс», 2004. -624 с.
16. Кубенский, А. А. Функциональное программирование : учебник и практикум для академического бакалавриата / А. А. Кубенский. — М. : Издательство Юрайт, 2019. — 348 с.
17. Кудрина, Е. В. Основы алгоритмизации и программирования на языке c# : учеб. пособие для бакалавриата и специалитета / Е. В. Кудрина, М. В. Огнева. — М. : Издательство Юрайт, 2019. — 322 с.
18. Малявко, А. А. Формальные языки и компиляторы : учеб. пособие для вузов / А. А. Малявко. — М. : Издательство Юрайт, 2018. — 429 с.

19. Мамонова, Т. Е. Информационные технологии. Лабораторный практикум : учеб. пособие для СПО / Т. Е. Мамонова. — М. : Издательство Юрайт, 2019. — 178 с.
20. Мартынюк, Ю. М. Методы программирования [Текст] : учебное пособие / Ю. М. Мартынюк, С. С. Гербут, В. С. Ванькова ; рец.: Е. Г. Торина, Е. А. Снижко ; ФГБОУ ВПО "Тульский государственный педагогический университет им. Л. Н. Толстого". - Тула : Изд-во ТГПУ им. Л. Н. Толстого, 2013. - 70 с
21. Петцольд Ч. Программирование для Microsoft Windows на C#. Пер. с англ.— М.: Издательско-торговый дом «Русская Редакция», 2002. – 576с.
22. Полное руководство по языку программирования C# 6.0 и платформе .NET 4.6 [Электронный ресурс]. – Режим доступа: //http://metanit.com/sharp/tutorial/(дата обращения: 29.09.2021).
23. Сайт компании Inteltelecom / О компании. - URL: https://www.inteltelecom.ru/about-company/ (Дата обращения: 10.12.2021).
24. Сайт компании OKTELL/Функции. -URL: https://oktell.ru/oktell/functions/ (Дата обращения: 10.12.2021).
25. Сорокин А.А. Объектно-ориентированное программирование: учебное пособие (курс лекций) / А.А. Сорокин; Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Северо-Кавказский федеральный университет», Министерство образования и науки Российской Федерации. – Ставрополь : СКФУ, 2014. – 174с. : ил.
26. Столбовский Д.Н. Основы разработки Web-приложений на ASP.NET [Электронный ресурс] / Д.Н. Столбовский. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 375 с.

27. Суслова И. А. МУ по выполнению и оформлению ВКР [Текст] / И. А. Суслова, Е. В. Чубаркова — Екатеринбург: ФГАОУ ВО «Рос. гос. проф.-пед. ун-т», 2014. — 41 с.
28. Хайруллин Р.С. Программирование на C#: учебное пособие / Р.С. Хайруллин. – Казань: Изд-во Казанск. гос. архитект.-строит. ун-та, 2017. – 153 с.
29. Хорев П. Б. Объектно-ориентированное программирование с примерами на C# [Текст]: учебное пособие / П. Б. Хорев — 3-е изд. — Москва: Форум, Инфра-М, 2016. — 200 с.
30. Черткова, Е. А. Статистика. Автоматизация обработки информации : учеб. пособие для вузов / Е. А. Черткова ; под общ. ред. Е. А. Чертковой. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2017. — 195 с.
31. Шилдт Г. Полное руководство C# 4.0 [Текст]: учебное пособие / Г. Шилдт — пер. с англ. Берштейн И. В. — Москва: Вильямс, 2012.— 1051 с.
32. Ю. А. Маглинец. Анализ требований к автоматизированным информационным системам. М.: Бином, 2008.

Приложения

Приложение 1.

Акт апробации и внедрения.

Приложение для автоматизации проверки знаний учащихся для МОУ СОШ №20.

Составлен «01» марта 2022 г.

В МОУ СОШ №20 г. Каменска-Уральского в рамках выпускной квалификационной работы «Приложение для автоматизации проверки знаний учащихся» была проведена апробация и внедрение программы.

В процессе апробации были выполнены следующие работы:

1. На каждый компьютер учителя установлен пакет ExamCreator.exe.
2. На каждый компьютер ученика установлен пакет Exam.exe.
3. Проведено обучение преподавателей и лаборанта по работе с программой.

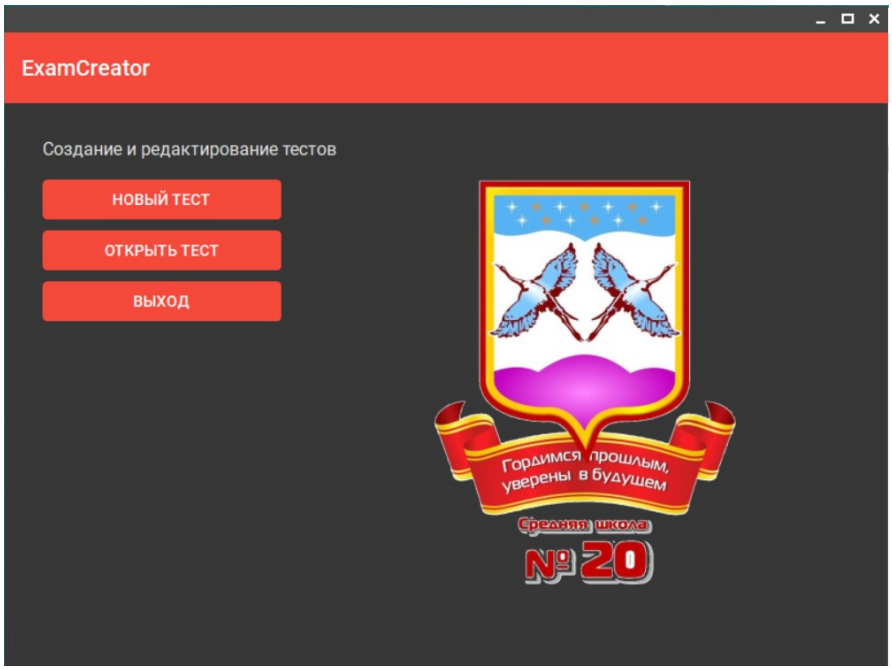
Заключение по результатам апробации и внедрения приложения для автоматизации.

1. Работы по внедрению и апробации приложения для автоматизации проверки знаний учащихся проведены в полном объеме.
2. Программа удобна в использовании, содержит необходимый функционал.

Директор МОУ СОШ №20 г. Каменска-Уральского

_____/ Щеголькова А.В.

Приложение 2.



Код программы приложения для учителей по созданию тестов.

Основное тело программы:

```
using System;
using MaterialSkin;
using MaterialSkin.Controls;

namespace ExamCreator.Forms
{
    /// <summary>
    /// Главная форма (Главное меню)
    /// </summary>
    public partial class MainForm : MaterialForm
    {
        /// <summary>
        /// Переменная для отслеживания открытия теста
        /// </summary>
        private bool _opening;


        /// <summary>
        /// Конструктор главной формы
        /// </summary>
        public MainForm()
        {
            InitializeComponent();

            // Настройка цветовой темы для окон
            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);
            materialSkinManager.Theme = MaterialSkinManager.Themes.DARK;
            materialSkinManager.ColorScheme = new ColorScheme(
```

```
                Primary.Red500,
                Primary.Grey800,
                Primary.Red900,
                Accent.Red700,
                TextShade.WHITE);
        }

        /// <summary>
        /// Нажатие на кнопку "Новый тест"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bNew_Click(object sender, EventArgs e)
        {
            // Создаем редактор для нового теста
            _opening = false;
            var editor = new Editor(ref _opening);

            // Отображаем едактор, скрываем меню
            editor.Show();
            Hide();
        }

        /// <summary>
        /// Нажатие на кнопку "Открыть тест"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bOpen_Click(object sender, EventArgs e)
        {
            // Создаем редактор для открытого теста
            _opening = true;
            var editor = new Editor(ref _opening);

            // Если пользователь прервал открытие, то выходим из функции
            if (!_opening)
            {
                return;
            }

            // Отображаем редактор, скрываем меню
            editor.Show();
            Hide();
        }

        /// <summary>
        /// Нажатие на кнопку "Выход"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bExit_Click(object sender, EventArgs e)
        {
            // Закрыть MainForm = закрыть все
            Close();
        }
    }
}
```

Создание формы «редактор»:

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using ExamCreator.Classes;
using MaterialSkin;
using MaterialSkin.Controls;

namespace ExamCreator.Forms
{
    /// <summary>
    /// Форма "Редактор"
    /// </summary>
    public partial class Editor : MaterialForm
    {
        /// <summary>
        /// Список страниц теста
        /// </summary>
        private readonly List<Page> _pages = new List<Page>();
        /// <summary>
        /// Индекс текущей страницы
        /// </summary>
        private int _currentIndex;
        /// <summary>
        /// Списки полей для ввода и чекбоксов
        /// </summary>
        private List<MaterialTextBox2> _textBoxes = new
List<MaterialTextBox2>();
        private List<MaterialCheckbox> _checkBoxes = new
List<MaterialCheckbox>();

        /// <summary>
        /// Конструктор редактора
        /// </summary>
        /// <param name="opening"></param>
        public Editor(ref bool opening)
        {
            InitializeComponent();

            // Применяем тему с главной формы
            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);

            // Добавляем поля ввода в список полей
            _textBoxes.Add(tQuestion);
            _textBoxes.Add(tAnswer1);
            _textBoxes.Add(tAnswer2);
            _textBoxes.Add(tAnswer3);
            _textBoxes.Add(tAnswer4);

            // добавляем чекбоксы в список
            _checkBoxes.Add(cbOne);
            _checkBoxes.Add(cbTwo);
            _checkBoxes.Add(cbThree);
            _checkBoxes.Add(cbFour);

            // Если происходит открытие теста для редактирования, то
            if (opening)
            {
                // Создаем объект класса открытия
```

```
                var open = new Opener(ref _pages, ref _textBoxes, ref
_checkBoxes, ref opening);
                // Если пользователь прервал открытие, то выходим из функции
                if (!opening)
                {
                    return;
                }

                // Обновляем полосу прогресса и надпись текущего вопроса
                pbQuestions.Maximum = _pages.Count;
                pbQuestions.Value = 1;
                lQuestionNum.Text = $@"Вопрос
{pbQuestions.Value}/{pbQuestions.Maximum}";
            }
            // Иначе происходит создание нового теста
            else
            {
                // Создаем новую страницу и добавляем ее в список страниц
                var page = new Page(tQuestion, tAnswer1, tAnswer2, tAnswer3,
tAnswer4, cbOne, cbTwo, cbThree, cbFour, gData);
                _pages.Add(page);

                // Обновляем полосу прогресса и надпись текущего вопроса
                pbQuestions.Maximum = _pages.Count;
                pbQuestions.Value = 1;
                lQuestionNum.Text = $@"Вопрос
{pbQuestions.Value}/{pbQuestions.Maximum}";
            }
        }

        /// <summary>
        /// Функция триггер при закрытии редактора
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Editor_FormClosed(object sender, FormClosedEventArgs e)
        {
            // Переходим в главную форму
            var mainForm = Application.OpenForms[0];
            mainForm.Show();
        }

        /// <summary>
        /// Нажатие кнопки "В главное меню"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bBack_Click(object sender, EventArgs e)
        {
            // Закрыть редактор
            Close();
        }

        /// <summary>
        /// Нажатие кнопки "Сохранить тест"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bSaveTest_Click(object sender, EventArgs e)
        {
```

```
            // Создаем объект класса сохранение
            var save = new Saver(_pages);
        }

        /// <summary>
        /// Нажатие кнопки "Сохранить вопрос"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bSaveQuestion_Click(object sender, EventArgs e)
        {
            // Перезаписываем страницу теста информацией со страницы редактора
            var page = new Page(tQuestion, tAnswer1, tAnswer2, tAnswer3,
tAnswer4, cbOne, cbTwo, cbThree, cbFour, gData);
            _pages[_currentIndex] = page;
        }

        /// <summary>
        /// Нажатие кнопки "Новый вопрос"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bNewQuestion_Click(object sender, EventArgs e)
        {
            // Отчищаем поля
            foreach (var tb in _textBoxes)
            {
                tb.Text = null;
            }

            // Отчищаем чекбоксы
            foreach (var cb in _checkBoxes)
            {
                cb.Checked = false;
            }


            // Создаем новую страницу и добавляем ее в список страниц
            var page = new Page(tQuestion, tAnswer1, tAnswer2, tAnswer3,
tAnswer4, cbOne, cbTwo, cbThree, cbFour, gData);
            _pages.Add(page);


            // Текущий индекс равен последней странице теста
            _currentIndex = _pages.Count - 1;

            // Обновляем полосу прогресса и надпись текущего вопроса
            pbQuestions.Maximum += 1;
            pbQuestions.Value = _pages.Count;
            lQuestionNum.Text = $@"Вопрос
{pbQuestions.Value}/{pbQuestions.Maximum}";
        }


        /// <summary>
        /// Нажатие на кнопку "Отчистить поля"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bClear_Click(object sender, EventArgs e)
        {
            // Отчищаем поля
            foreach (var tb in _textBoxes)
```

```
            {
                tb.Text = null;
            }

            // Отчищаем чекбоксы
            foreach (var cb in _checkBoxes)
            {
                cb.Checked = false;
            }
        }

        /// <summary>
        /// Нажатие на кнопку "Удалить вопрос"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bDelQuestion_Click(object sender, EventArgs e)
        {
            // Отчищаем поля
            foreach (var tb in _textBoxes)
            {
                tb.Text = null;
            }

            // Отчищаем чекбоксы
            foreach (var cb in _checkBoxes)
            {
                cb.Checked = false;
            }

            // Если кол-во страниц = 1, то выходим из функции
            if (_pages.Count == 1) return;


            // Удаляем страницу из списка страниц
            _pages.RemoveAt(_currentIndex);

            // Обновляем полосу прогресса и надпись текущего вопроса
            bPrev.PerformClick();
            pbQuestions.Maximum -= 1;
            lQuestionNum.Text = $@"Вопрос
{pbQuestions.Value}/{pbQuestions.Maximum}";

        }

        /// <summary>
        /// Нажатие кнопки "Предыдущий вопрос"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bPrev_Click(object sender, EventArgs e)
        {
            // Если пользователь на первой странице, то выходим из функции
            if (_currentIndex == 0) return;


            // Уменьшаем текущий индекс на 1
            _currentIndex -= 1;
            // Загружаем предыдущую страницу
            var load = new Loader(_pages, _currentIndex, ref _textBoxes, ref
_checkBoxes);
```

```
            // Обновляем полосу прогресса и надпись текущего вопроса
            pbQuestions.Value -= 1;
            lQuestionNum.Text = $@"Вопрос
{pbQuestions.Value}/{pbQuestions.Maximum}";
        }

        /// <summary>
        /// Нажатие кнопки "Следующий вопрос"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bNext_Click(object sender, EventArgs e)
        {
            // Если пользователь на последней странице, то выходим из функции
            if (_currentIndex == _pages.Count - 1) return;

            // Увеличиваем текущий индекс на 1
            _currentIndex += 1;
            // Загружаем следующую страницу
            var load = new Loader(_pages, _currentIndex, ref _textBoxes, ref
_checkBoxes);

            // Обновляем полосу прогресса и надпись текущего вопроса
            pbQuestions.Value += 1;
            lQuestionNum.Text = $@"Вопрос
{pbQuestions.Value}/{pbQuestions.Maximum}";
        }
    }
}
```

Создание класса «шифровальщик»:

```
using System.IO;

namespace ExamCreator.Classes
{
    /// <summary>
    /// Класс шифровальщик
    /// </summary>
    public class Cryptographer
    {
        /// <summary>
        /// Ключ шифрования
        /// </summary>
        private const byte Key = 111;

        /// <summary>
        /// Стандартный конструктор
        /// </summary>
        /// <param name="filename"></param>
        public Cryptographer(string filename)
        {
            // Функция шифрования XOR
            byte[] Crypt(byte[] bytes)
            {
                for (var i = 0; i < bytes.Length; i++)
                    bytes[i] ^= Key;
                return bytes;
            }
```

```
            // Функция определения нового расширения файла
            string NewFileName(string fileName)
            {
                // Если файл уже с расширением .crypt, то возвращаем имя файла
                if (fileName.EndsWith(".crypt")) return fileName;
                fileName = Path.ChangeExtension(fileName, "crypt");
                return fileName;
            }

            // Загружаем файл теста .xml
            var myFile = File.ReadAllBytes(filename);
            // Зашифровываем файл
            var newFile = Crypt(myFile);
            // Определяем новое расширение
            var newFileName = NewFileName(filename);
            // Сохраняем зашифрованный файл с новым расширением
            File.WriteAllBytes(newFileName, newFile);
        }
    }
}
```

Создание класса «загрузчик»:

```
using System;

using System.Collections.Generic;

using System.Linq;

using MaterialSkin.Controls;



namespace ExamCreator.Classes

{

    /// <summary>

    /// Класс загрузчика

    /// </summary>

    public class Loader

    {

        /// <summary>

        /// Стандартный конструктор

        /// </summary>

        /// <param name="pages"></param>
```

```
        /// <param name="currentIndex"></param>

        /// <param name="textBoxes"></param>

        /// <param name="checkBoxes"></param>

        public Loader(IReadOnlyList<Page> pages, int currentIndex, ref
List<MaterialTextBox2> textBoxes, ref List<MaterialCheckbox> checkBoxes)

        {

            // Определяем страницу теста под текущим индексом из списка страниц

            var page = pages[currentIndex];



            // Записываем данные со страницы в поля редактора

            for (var i = 0; i < textBoxes.Count; i++)

            {

                switch (i)

                {

                    case 0:

                        textBoxes[i].Text = page.Question;

                        break;

                    case 1:

                        textBoxes[i].Text = page.Answer1;

                        break;

                    case 2:

                        textBoxes[i].Text = page.Answer2;

                        break;

                    case 3:

                        textBoxes[i].Text = page.Answer3;

                        break;

                    case 4:

                        textBoxes[i].Text = page.Answer4;
```

```
                        break;
                }
            }

            // Отмечаем чекбоксы со страницы в чекбоксы редактора
            foreach (var cb in checkBoxes)
            {
                // Сначала отчистим все
                cb.Checked = false;

                // Отметим верные
                foreach (var index in page.Correct.Where(index =>
Convert.ToInt32(cb.Tag) == index))
                {
                    cb.Checked = true;
                }
            }
        }
    }
}
```

Создание класса «открытие»:

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Xml;
using MaterialSkin.Controls;

namespace ExamCreator.Classes
```

```
{
    /// <summary>
    /// Класс открытия
    /// </summary>
    public class Opener
    {
        /// <summary>
        /// Стандартный конструктор
        /// </summary>
        /// <param name="pages"></param>
        /// <param name="textBoxes"></param>
        /// <param name="checkBoxes"></param>
        /// <param name="opening"></param>
        public Opener(ref List<Page> pages, ref List<MaterialTextBox2>
textBoxes, ref List<MaterialCheckbox> checkBoxes, ref bool opening)
        {
            // Отчищаем список страниц
            pages.Clear();

            // Диалог с пользователем "Открыть"
            var openDialog = new OpenFileDialog();
            openDialog.Filter = @"Test file(*.xml)|*.xml|All files(*.*)|*.*";

            // Если пользователь прервал операцию, то вернемся в главное меню
            if (openDialog.ShowDialog() == DialogResult.Cancel)
            {
                opening = false;
                return;
```

```
}

// Полное имя файла
var filename = openDialog.FileName;

// Определим и загрузим открываемый xml файл
var xDoc = new XmlDocument();
xDoc.Load(filename);

// Определим корневой элемент xml файла
var xRoot = xDoc.DocumentElement;
// Если xml-файл не пустой, то перебираем файл
if (xRoot != null)
{
    foreach (XmlElement xNode in xRoot)
    {
        // Определяем новую страницу теста
        var page = new Page();

        foreach (XmlNode cNode in xNode.ChildNodes)
        {
            switch (cNode.Name)
            {
                case "Question":
                    page.Question = cNode.InnerText;
                    break;
                case "Answer1":
```

```
                                page.Answer1 = cNode.InnerText;
                                break;
                            case "Answer2":
                                page.Answer2 = cNode.InnerText;
                                break;
                            case "Answer3":
                                page.Answer3 = cNode.InnerText;
                                break;
                            case "Answer4":
                                page.Answer4 = cNode.InnerText;
                                break;
                            case "Correct":
                            {
                                foreach (XmlElement cNodeCorrect in
cNode.ChildNodes)
                                {
page.Correct.Add(Convert.ToInt32(cNodeCorrect.InnerText));
                                }

                                break;
                            }
                        }
                    }

                    // Добавляем страницу в список страниц
                    pages.Add(page);
                }
```

```
            }
            // Иначе добавляем в список страниц пустую страницу
            else
            {
                pages.Add(new Page());
            }

            // Загружаем первую страницу в редактор
            var load = new Loader(pages, 0, ref textBoxes, ref checkBoxes);
        }
    }
}
```

Создание класса «страница теста»:

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using MaterialSkin.Controls;

namespace ExamCreator.Classes
{
    /// <summary>
    /// Класс страницы теста
    /// </summary>
    public class Page
    {
        /// <summary>
        /// Вопрос на странице
```

```
/// </summary>
public string Question;
/// <summary>
/// Ответ 1
/// </summary>
public string Answer1;
/// <summary>
/// Ответ 2
/// </summary>
public string Answer2;
/// <summary>
/// Ответ 3
/// </summary>
public string Answer3;
/// <summary>
/// Ответ 4
/// </summary>
public string Answer4;
/// <summary>
/// Список тэгов верных ответов
/// </summary>
public readonly List<int> Correct = new List<int>();

/// <summary>
/// Пустой конструктор для XmlSerializer
/// </summary>
public Page() {}
```

```
/// <summary>
/// Стандартный конструктор
/// </summary>
/// <param name="tQuestion"></param>
/// <param name="tAnswer1"></param>
/// <param name="tAnswer2"></param>
/// <param name="tAnswer3"></param>
/// <param name="tAnswer4"></param>
/// <param name="cbOne"></param>
/// <param name="cbTwo"></param>
/// <param name="cbThree"></param>
/// <param name="cbFour"></param>
/// <param name="gData"></param>
public Page(MaterialTextBox2 tQuestion,
    MaterialTextBox2 tAnswer1,
    MaterialTextBox2 tAnswer2,
    MaterialTextBox2 tAnswer3,
    MaterialTextBox2 tAnswer4,
    MaterialCheckbox cbOne,
    MaterialCheckbox cbTwo,
    MaterialCheckbox cbThree,
    MaterialCheckbox cbFour,
    GroupBox gData)
{
    // Записываем значения из полей ввода в поля объекта
    Question = tQuestion.Text;
    Answer1 = tAnswer1.Text;
```

```
            Answer2 = tAnswer2.Text;
            Answer3 = tAnswer3.Text;
            Answer4 = tAnswer4.Text;
            // Записываем значения из чекбоксов в список верных ответов
            foreach (var item in gData.Controls)
            {
                if (item is MaterialCheckbox cb && cb.Checked)
                {
                    Correct.Add(Convert.ToInt32(cb.Tag));
                }
            }
        }
    }
}
```

Создание класса «сохранение»:

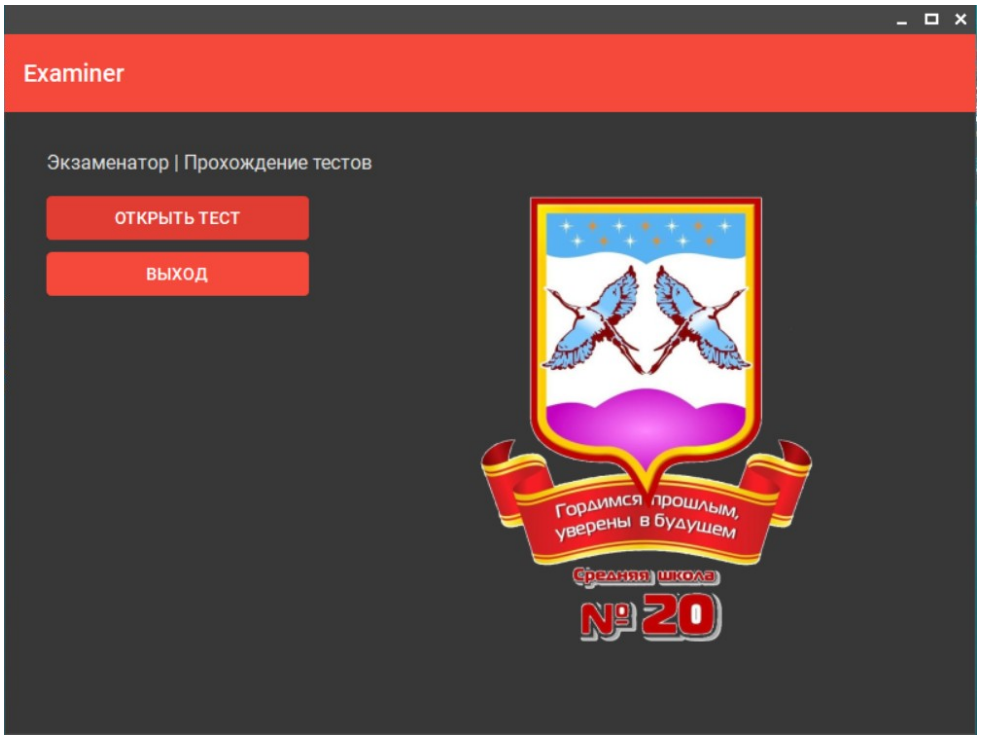
```
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;
using System.Xml.Serialization;

namespace ExamCreator.Classes
{
    /// <summary>
    /// Класс сохранения
    /// </summary>
    public class Saver
    {
```

```
        /// <summary>
        /// Стандартный конструктор
        /// </summary>
        /// <param name="pages"></param>
        public Saver(List<Page> pages)
        {
            // Диалог с пользователем "Сохранить как"
            var saveDialog = new SaveFileDialog();
            saveDialog.Filter = @"Test file(*.xml)|*.xml|All files(*.*)|*.*";
            // Если пользователь прервал операцию, то выходим из конструктора
            if (saveDialog.ShowDialog() == DialogResult.Cancel)
            {
                return;
            }
            // Полное имя файла
            var filename = saveDialog.FileName;
            // Сериализация xml (Сохраняем список страниц теста в xml файл)
            var formatter = new XmlSerializer(typeof(Page[]));
            using (var fs = new FileStream(filename, FileMode.OpenOrCreate))
            {
                var temp = pages.ToArray();
                formatter.Serialize(fs, temp);
            }
            // Шифрование файла теста
            var cryptographer = new Cryptographer(filename);
        }
    }
```

```
}
```

Приложение 3.



Код создания приложения для учеников для прохождения тестирования.

Основное тело программы:

```
using System;
using MaterialSkin;
using MaterialSkin.Controls;


namespace Examiner.Forms
{
    /// <summary>
    /// Главная форма (Главное меню)
    /// </summary>
    public partial class MainForm : MaterialForm
    {
```

```
/// <summary>
/// Переменная для отслеживания открытия теста
/// </summary>
private bool _opening;

/// <summary>
/// Конструктор главной формы
/// </summary>
public MainForm()
{
    InitializeComponent();

    // Настройка цветовой темы для окон
    var materialSkinManager = MaterialSkinManager.Instance;
    materialSkinManager.AddFormToManage(this);
    materialSkinManager.Theme = MaterialSkinManager.Themes.DARK;
    materialSkinManager.ColorScheme = new ColorScheme(
        Primary.Red500,
        Primary.Grey800,
        Primary.Red900,
        Accent.Red700,
        TextShade.WHITE);
}

/// <summary>
/// Нажатие на кнопку "Открыть тест"
/// </summary>
```

```
/// <param name="sender"></param>
/// <param name="e"></param>
private void bOpen_Click(object sender, EventArgs e)
{
    // Создаем экзаменатор для открытого теста
    _opening = true;
    var tester = new Tester(ref _opening);

    // Если пользователь прервал открытие, то выходим из функции
    if (!_opening)
    {
        return;
    }

    // Отображаем экзаменатор, скрываем меню
    tester.Show();
    Hide();
}

/// <summary>
/// Нажатие на кнопку "Выход"
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void bExit_Click(object sender, EventArgs e)
{
    Close();
```

```
        }
    }
}
```

Создание формы «экзаменатор»:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Examiner.Classes;
using MaterialSkin;
using MaterialSkin.Controls;

namespace Examiner.Forms
{
    /// <summary>
    /// Форма "Экзаменатор"
    /// </summary>
    public partial class Tester : MaterialForm
    {
        /// <summary>
        /// Список страниц теста
        /// </summary>
        private readonly List<Page> _pages = new List<Page>();
        /// <summary>
        /// Индекс текущей страницы
        /// </summary>
        private int _currentIndex;
```

```
        /// <summary>
        /// Список надписей
        /// </summary>
        private List<MaterialLabel> _labels = new List<MaterialLabel>();
        /// <summary>
        /// Список чекбоксов
        /// </summary>
        private List<MaterialCheckbox> _checkBoxes = new
List<MaterialCheckbox>();
        /// <summary>
        /// Список из списков отмеченных ответов
        /// </summary>
        private readonly List<List<int>> _answers = new List<List<int>>();

        /// <summary>
        /// Стандартный конструктор
        /// </summary>
        /// <param name="opening"></param>
        public Tester(ref bool opening)
        {
            InitializeComponent();

            // Применяем тему с главной формы
            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);

            // Добавляем надписи в список надписей
            _labels.Add(lQuestion);
```

```
            _labels.Add(lAnswer1);
            _labels.Add(lAnswer2);
            _labels.Add(lAnswer3);
            _labels.Add(lAnswer4);
            _labels.Add(lAnswers);

            // Добавляем чекбоксы в список чекбоксы
            _checkBoxes.Add(cbOne);
            _checkBoxes.Add(cbTwo);
            _checkBoxes.Add(cbThree);
            _checkBoxes.Add(cbFour);

            // Если происходит открытие теста для прохождения, то
            if (opening)
            {
                // Создаем объект класса открытия
                var open = new Opener(ref _pages, ref _labels, ref _checkBoxes,
ref opening);
                // Если пользователь прервал открытие, то выходим из функции
                if (!opening)
                {
                    return;
                }

                // Обновляем полосу прогресса и надпись текущего вопроса
                pbQuestions.Maximum = _pages.Count;
                pbQuestions.Value = 1;
```

```
                lQuestionNum.Text = $@"Вопрос
{pbQuestions.Value}/{pbQuestions.Maximum}";

                // Определяем длину списка ответов, основываясь на длине теста

                _answers = new List<List<int>>(_pages.Count);


                // Определяем все ответы сначала как отрицательные

                for (var i = 0; i < _answers.Capacity; i++)

                {

                    _answers.Add(new List<int>());

                }

            }

            // Иначе выводим ошибку

            else

            {

                MessageBox.Show("Ошибка в чтении файла!");

                Close();

            }

        }


        /// <summary>

        /// Функция триггер при закрытии экзаменатора

        /// </summary>

        /// <param name="sender"></param>

        /// <param name="e"></param>

        private void Tester_FormClosed(object sender, FormClosedEventArgs e)

        {

            var mainForm = Application.OpenForms[0];

            mainForm.Show();
```

```
}

/// <summary>
/// Нажатие кнопки "В главное меню"
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void bBack_Click(object sender, EventArgs e)
{
    // Закрыть экзаменатор
    Close();
}
/// <summary>
/// Нажатие кнопки "Завершить тест"
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void bEndTest_Click(object sender, EventArgs e)
{
    // Создаем объект класса проверки теста
    var saver = new Saver(_pages, _answers);
    // Закрываем экзаменатор
    Close();
}

/// <summary>
/// Нажатие кнопки "Предыдущий вопрос"
```

```
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bPrev_Click(object sender, EventArgs e)
        {
            // Если пользователь на первой странице, то выходим из функции
            if (_currentIndex == 0) return;

            // Уменьшаем текущий индекс на 1
            _currentIndex -= 1;
            // Загружаем предыдущую страницу
            var load = new Loader(_pages, _currentIndex, ref _labels, ref
_checkBoxes, _answers);

            // Обновляем полосу прогресса и надпись текущего вопроса
            pbQuestions.Value -= 1;
            lQuestionNum.Text = $@"Вопрос
{pbQuestions.Value}/{pbQuestions.Maximum}";
        }

        /// <summary>
        /// Нажатие кнопки "Следующий вопрос"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bNext_Click(object sender, EventArgs e)
        {
            // Если пользователь на последней странице, то выходим из функции
```

```
            if (_currentIndex == _pages.Count - 1) return;



            // Увеличиваем текущий индекс на 1

            _currentIndex += 1;

            // Загружаем следующую страницу

            var load = new Loader(_pages, _currentIndex, ref _labels, ref
_checkBoxes, _answers);



            // Обновляем полосу прогресса и надпись текущего вопроса

            pbQuestions.Value += 1;

            lQuestionNum.Text = $@"Вопрос
{pbQuestions.Value}/{pbQuestions.Maximum}";

        }



        /// <summary>

        /// Нажатие на кнопку "Сохранить ответ"

        /// </summary>

        /// <param name="sender"></param>

        /// <param name="e"></param>

        private void bSaveAnswer_Click(object sender, EventArgs e)

        {

            // Отчищаем текущие ответы на вопрос

            _answers[_currentIndex].Clear();



            // Сохраняем ответы со страницы экзаменатора

            foreach (var cb in _checkBoxes.Where(cb => cb.Checked))

            {

                _answers[_currentIndex].Add(Convert.ToInt32(cb.Tag));
```

```
            }
        }
    }
}
```

Создание класса «шифровальщик»:

```
using System.IO;
namespace Examiner.Classes
{
    /// <summary>
    /// Класс шифровальщик
    /// </summary>
    public class Cryptographer
    {
        /// <summary>
        /// Ключ шифрования
        /// </summary>
        private const byte Key = 111;
        /// <summary>
        /// Стандартный конструктор
        /// </summary>
        /// <param name="filename"></param>
        public Cryptographer(string filename)
        {
            // Функция расшифрования XOR
            byte[] Crypt(byte[] bytes)
            {
                for (var i = 0; i < bytes.Length; i++)
```

```
            bytes[i] ^= Key;
        return bytes;
    }
    // Функция определения нового расширения файла
    string NewFileName(string fileName)
    {
        // Если файл уже с расширением .xml, то возвращаем имя файла
        if (fileName.EndsWith(".xml")) return fileName;
        fileName = Path.ChangeExtension(fileName, "xml");
        return fileName;
    }
    // Загружаем файл теста .crypt
    var myFile = File.ReadAllBytes(filename);
    // Расшифровываем файл
    var newFile = Crypt(myFile);
    // Временный каталог хранения расшифрованного теста
    const string path = @"C:\temp\Examiner";
    // Если каталога не существует, то
    if (!Directory.Exists(path))
    {
        // Создаем каталог
        Directory.CreateDirectory(path);
    }
    // Определяем новое расширение
    var newFileName = NewFileName(path + @"\tmp.xml");
// Сохраняем расшифрованный файл с новым расширением
    File.WriteAllBytes(newFileName, newFile);
```

```
            }
        }
    }
```

Создание класса «загрузчик»:

```
using System;
using System.Collections.Generic;
using System.Linq;
using MaterialSkin.Controls;

namespace Examiner.Classes
{
    /// <summary>
    /// Класс загрузчика
    /// </summary>
    public class Loader
    {
        /// <summary>
        /// Стандартный конструктор
        /// </summary>
        /// <param name="pages"></param>
        /// <param name="currentIndex"></param>
        /// <param name="labels"></param>
        /// <param name="checkBoxes"></param>
        /// <param name="answers"></param>
              public Loader(IReadOnlyList<Page> pages, int currentIndex, ref
List<MaterialLabel>    labels,    ref    List<MaterialCheckbox>    checkBoxes,
IReadOnlyList<List<int>> answers = null)
        {
            // Определяем страницу теста под текущим индексом из списка страниц
            var page = pages[currentIndex];

            // Записываем данные со страницы в надписи экзаменатора
            for (var i = 0; i < labels.Count; i++)
            {
                switch (i)
                {
                    case 0:
                        labels[i].Text = page.Question;
                        break;
                    case 1:
                        labels[i].Text = page.Answer1;
                        break;
                    case 2:
                        labels[i].Text = page.Answer2;
                        break;
                    case 3:
                        labels[i].Text = page.Answer3;
                        break;
                    case 4:
                        labels[i].Text = page.Answer4;
                        break;
                    case 5:
                         // Если количество правильных ответов больше одного, то
выводим соответствующую надпись, иначе для одного ответа
                            labels[i].Text = page.Correct.Count > 1 ? "Выберите
несколько верных вариантов ответа" : "Выберите верный вариант ответа";
                        break;
                }
            }

              // Отмечаем чекбоксы со страницы ответов пользователя в чекбоксы
экзаменатора
            foreach (var cb in checkBoxes)
```

```
            {
                // Сначала отчистим все
                cb.Checked = false;

                  // Если пользователь еще не давал ответов на этот вопрос, то
пропускаем итерацию
                if (answers?[currentIndex] == null) continue;
                // Выводим ответы пользователя
                   foreach (var answer in answers[currentIndex].Where(answer =>
Convert.ToInt32(cb.Tag) == answer))
                {
                    cb.Checked = true;
                }
            }
        }
    }
}
```

Создание класса «открытие»:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;
using System.Xml;
using MaterialSkin.Controls;

namespace Examiner.Classes
{
    /// <summary>
    /// Класс открытия
    /// </summary>
    public class Opener
    {
        /// <summary>
        /// Стандартный конструктор
        /// </summary>
        /// <param name="pages"></param>
        /// <param name="labels"></param>
        /// <param name="checkBoxes"></param>
        /// <param name="opening"></param>
         public Opener(ref List<Page> pages, ref List<MaterialLabel> labels, ref
List<MaterialCheckbox> checkBoxes, ref bool opening)
        {
            // Отчищаем список страниц
            pages.Clear();

            // Диалог с пользователем "Открыть"
            var openDialog = new OpenFileDialog();
               openDialog.Filter = @"Test file(*.crypt)|*.crypt|All files(*.*)|
*.*";


            // Если пользователь прервал операцию, то вернемся в главное меню
            if (openDialog.ShowDialog() == DialogResult.Cancel)
            {
                opening = false;
                return;
            }

            // Полное имя файла
            var filename = openDialog.FileName;
```

```
            // Определяем шифровальщик и расшифровываем файл
            var cryptographer = new Cryptographer(filename);

            // Временный каталог хранения теста
            const string path = @"C:\temp\Examiner";
            if (!Directory.Exists(path))
            {
                Directory.CreateDirectory(path);
            }

            // Полное имя файла
            filename = path + @"\tmp.xml";

            // Определим и загрузим открываемый xml файл
            var xDoc = new XmlDocument();
            xDoc.Load(filename);

            // Определим корневой элемент xml файла
            var xRoot = xDoc.DocumentElement;
            // Если xml-файл не пустой, то перебираем файл
            if (xRoot != null)
            {
                foreach (XmlElement xNode in xRoot)
                {
                    // Определяем новую страницу теста
                    var page = new Page();

                    foreach (XmlNode cNode in xNode.ChildNodes)
                    {
                        switch (cNode.Name)
                        {
                            case "Question":
                                page.Question = cNode.InnerText;
                                break;
                            case "Answer1":
                                page.Answer1 = cNode.InnerText;
                                break;
                            case "Answer2":
                                page.Answer2 = cNode.InnerText;
                                break;
                            case "Answer3":
                                page.Answer3 = cNode.InnerText;
                                break;
                            case "Answer4":
                                page.Answer4 = cNode.InnerText;
                                break;
                            case "Correct":
                                {
                                    foreach (XmlElement cNodeCorrect in
cNode.ChildNodes)
                                    {
                                        page.Correct.Add(Convert.ToInt32(cNodeCorrec
t.InnerText));
                                    }

                                    break;
                                }
                        }
```

```
                    }

                    // Добавляем страницу в список страниц
                    pages.Add(page);
                }
            }
            // Иначе добавляем в список страниц пустую страницу
            else
            {
                pages.Add(new Page());
            }

            // Загружаем первую страницу в редактор
            var load = new Loader(pages, 0, ref labels, ref checkBoxes);
        }
    }
}
```

Создание класса «страница теста»:

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using MaterialSkin.Controls;

namespace Examiner.Classes
```

```
{
    /// <summary>
    /// Класс страницы теста
    /// </summary>
    public class Page
    {
        /// <summary>
        /// Вопрос на странице
        /// </summary>
        public string Question;
        /// <summary>
        /// Ответ 1
        /// </summary>
        public string Answer1;
        /// <summary>
        /// Ответ 2
        /// </summary>
        public string Answer2;
        /// <summary>
        /// Ответ 3
        /// </summary>
        public string Answer3;
        /// <summary>
        /// Ответ 4
```

```
/// </summary>
public string Answer4;
/// <summary>
/// Список тэгов верных ответов
/// </summary>
public readonly List<int> Correct = new List<int>();


/// <summary>
/// Пустой конструктор для XmlSerializer
/// </summary>
public Page() {}

// /// <summary>
// /// Стандартный конструктор
// /// </summary>
// /// <param name="lQuestion"></param>
// /// <param name="lAnswer1"></param>
// /// <param name="lAnswer2"></param>
// /// <param name="lAnswer3"></param>
// /// <param name="lAnswer4"></param>
// /// <param name="cbOne"></param>
// /// <param name="cbTwo"></param>
// /// <param name="cbThree"></param>
// /// <param name="cbFour"></param>
```

```
// /// <param name="gData"></param>
// public Page(MaterialLabel lQuestion,
//     MaterialLabel lAnswer1,
//     MaterialLabel lAnswer2,
//     MaterialLabel lAnswer3,
//     MaterialLabel lAnswer4,
//     MaterialCheckbox cbOne,
//     MaterialCheckbox cbTwo,
//     MaterialCheckbox cbThree,
//     MaterialCheckbox cbFour,
//     GroupBox gData)
// {
//     // Записываем значения из полей ввода в поля объекта
//     Question = lQuestion.Text;
//     Answer1 = lAnswer1.Text;
//     Answer2 = lAnswer2.Text;
//     Answer3 = lAnswer3.Text;
//     Answer4 = lAnswer4.Text;
//
//     // Записываем значения из чекбоксов в список верных ответов
//     foreach (var item in gData.Controls)
//     {
//         if (item is MaterialCheckbox cb && cb.Checked)
//         {
```

```
        //              Correct.Add(Convert.ToInt32(cb.Tag));
        //          }
        //      }
        // }
    }
}
```

Создание класса «проверки теста»:

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace Examiner.Classes
{
    /// <summary>
    /// Класс проверки теста
    /// </summary>
    public class Saver
    {
        /// <summary>
        /// Набранные баллы
        /// </summary>
        private readonly int _score = 0;
        /// <summary>
        /// Максимально возможные баллы
```

```
/// </summary>
private readonly int _maxScore = 0;
/// <summary>
/// Временная переменная верных ответов
/// </summary>
private readonly int _tempScore = 0;
/// <summary>
/// Кол-во полных верных ответов на вопросы
/// </summary>
private readonly int _scoreQuestions = 0;
/// <summary>
/// Максимально возможное кол-во полных верных ответов на вопросы
/// </summary>
private readonly int _maxScoreQuestions = 0;


/// <summary>
/// Стандартный конструктор
/// </summary>
/// <param name="pages"></param>
/// <param name="answers"></param>
public Saver(List<Page> pages, IReadOnlyList<List<int>> answers)
{
    // Подсчет максимально возможных баллов в тесте
    foreach (var page in pages)
```

```
            {
                foreach (var cb in page.Correct)
                {
                    _maxScore += 1;
                }
            }

            // Подсчет набранных баллов пользователем в тесте
            for (var i = 0; i < answers.Count; i++)
            {
                _tempScore = 0;
                for (var j = 1; j <= 4; j++)
                {
                    if (answers[i].Contains(j) && pages[i].Correct.Contains(j))
                    {
                        _tempScore += 1;
                    }
                    else
                    {
                                                    if  (!answers[i].Contains(j)  ||
pages[i].Correct.Contains(j)) continue;
                        _tempScore = 0;
                        break;
                    }
                }
```

```
                _score += _tempScore;
            }

            // Обнуляем временную переменную
            _tempScore = 0;
             // Подсчет кол-во полных верных ответов на вопросы пользователем в
тесте
            for (var i = 0; i < answers.Count; i++)
            {
                _tempScore += 1;
                for (var j = 1; j <= 4; j++)
                {
                      if (answers[i].Contains(j) && pages[i].Correct.Contains(j)
|| !answers[i].Contains(j) && !pages[i].Correct.Contains(j))
                    {
                    }
                    else
                    {
                        _tempScore -= 1;
                        break;
                    }
                }
            }
            // Кол-во полных верных ответов на вопросы пользователем
            _scoreQuestions = _tempScore;
             // Определяем максимально возможное кол-во полных верных ответов на
вопросы
            _maxScoreQuestions = pages.Count;

            // Выводим сообщение о кол-ве набранных баллов пользователю
            MessageBox.Show($"Кол-во набранных баллов: {_score}/{_maxScore}\n" +
                                $"Процент: {Math.Round(Convert.ToDouble(_score) /
Convert.ToDouble(_maxScore) * 100, 2)}%\n" +
                                    $"Кол-во полных верных ответов на вопросы:
{_scoreQuestions}/{_maxScoreQuestions}\n" +
                                                                        $"Процент:
{Math.Round(Convert.ToDouble(_scoreQuestions)                                   /
Convert.ToDouble(_maxScoreQuestions) * 100, 2)}%");
        }
    }
}
```