Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Основы алгоритмизации и программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**КОНСТРУКТОР ЭКЗАМЕНАЦИОННЫХ БИЛЕТОВ**

БГУИР КР 6 - 05 - 06 12 01 029 ПЗ

Выполнил: студент группы 310901Усов А. М.

Проверил: Кабариха В. А.

Минск 2024

Содержание

[**ВВЕДЕНИЕ 3**](#_Toc161594816)

[**1. ТРЕБОВАНИЯ К ПРОГРАММЕ 4**](#_Toc161594817)

[**1.1 Описание предметной области 4**](#_Toc161594818)

[**1.2 Системные требования 4**](#_Toc161594819)

[**1.3 Функциональные требования 5**](#_Toc161594820)

[**2. Конструирование программы 7**](#_Toc161594821)

[**2.1 Описание модулей программы 7**](#_Toc161594822)

[**2.2 Выбор способа организации данных 8**](#_Toc161594823)

[**2.3 Разработка перечня пользовательских функций программы 11**](#_Toc161594824)

ВВЕДЕНИЕ

Традиционно подготовка экзаменационных билетов была крайне трудоемким и длительным процессом для преподавателей. Им приходилось вручную составлять сотни вопросов по пройденному материалу, распечатывать их, компоновать в билеты и следить за актуальностью вопросов из года в год. Этот процесс занимал колоссальное количество времени и требовал серьезных усилий.

Современные тенденции заключаются в разработке специализированных приложений-конструкторов для максимальной автоматизации процесса на всех этапах – от создания шаблонов билетов и банка вопросов до формирования комплектов билетов и их печати. Такие решения способны существенно экономить время преподавателей, снижать вероятность ошибок, обеспечивать актуальность материалов.

Однако существующие программные продукты часто обладают ограниченным функционалом, плохо интегрируются с другими системами учебного заведения, имеют неудобный и запутанный интерфейс. Зачастую они решают лишь локальные задачи автоматизации, в то время как комплексный подход отсутствует.

Цель данной курсовой работы – создание программного обеспечения "Конструктор экзаменационных билетов" для автоматизации процесса подготовки экзаменационных материалов в учебных заведениях.

Для достижения поставленной цели были сформулированы следующие задачи:

* провести анализ требований к экзаменационным билетам в различных учебных заведениях;
* изучить существующие программные решения, выявить их достоинства и недостатки;
* спроектировать архитектуру приложения с учетом требований масштабируемости и расширяемости;
* реализовать функционал по созданию билетов и формированию банка вопросов;
* обеспечить возможность генерации комплектов билетов по заданным правилам;
* предусмотреть различные варианты вывода подготовленных билетов (печать, электронный формат).

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1 Описание предметной области

Учебные компьютерные программы, такие как «Конструктор экзаменационных билетов», позволяют моделировать процесс создания, проверки и экспорта экзаменационных билетов.

В разрабатываемой системе главными объектами являются экзаменационные билеты и тесты, рейтинг, пользователи.

Билеты включают в себе вопросы, дату создания, учреждение, тему вопросов и изготовитель билетов. Тесты представляют собой набор вопросов определенных категорий, также содержит информацию про автора, сложность, и сам набор вопросов. Рейтинг включат в себе тест, к которому относится рейтинг, и набор значений типа пользователь и его рейтинг по этому тесту. Пользователи включает в себя информацию об определенном пользователе, включая имя, ID, возраст, учебный класс и список созданных тестов.

1.2 Системные требования

Для выполнение курсовой роботы Конструктор экзаменационных билетов была выбрана популярная среда разработки Microsoft Visual Studio. Её преимущества включают в себя такие важные возможности, как хорошая обширная документация, удобный и понятный интерфейс.

В качестве языка программирования был выбран язык C++, который позволит при разработке программного обеспечения конструктор экзаменационных билетов гибкое управление памятью и скорость работы.

Для реализации интерфейса было выбрана реализация через консоль. Он обеспечивает простую реализацию для разработчика и понятный интерфейс.

Парадигма программирования была выбрана процедурная парадигма программирования, так как она обеспечивает хорошую структуру реализации, простоту для чтения кода и обеспечивает эффективность разработки.

Для реализации хранения данных была выбрана файловая структура, которая обеспечивает простое взаимодействие с языком программирования, простое редактирование при необходимости. Структура каждого файла представляет собой реализацию каждого класса.

1.3 Функциональные требования

При создании программного средства для автоматизации создания билетов к экзамену определяются функциональные требования, которые необходимо внедрить в систему. Такие как:

* возможность регистрации новых пользователей,
* авторизации уже зарегистрированных пользователей
* авторизация в качестве гостя;
* прохождение выбранного теста;
* возможность просмотра рейтинга пользователей, прошедших определенный тест;
* процесс экспорта созданного теста на печать;
* генерация билетов;
* создание и редактирование вопросов и ответов;
* процесс автоматической оценки после прохождения теста.
* возможность сортировать тесты по сложности;
* возможность фильтровать тесты по разным параметрам;
* гость должен иметь возможность пройти любой тест из списка публичных тестов.

Пользователь должен иметь возможность проходить публичные тесты, а также создавать свои собственные тесты с возможностью из редактирования.

Администратор должен иметь возможность делать все выше перечисленное, а также редактирование любых тестов и возможность менять рейтинг.

1.4 Описание основных нефункциональных требований

При выполнении курсовой работы «Конструктор экзаменационных билетов» для создания безопасного и надежного программного обеспечения были поставлены следующие нефункциональные требования:

* все хранимые данные должны быть представлены в виде шифра по определенному алгоритму;
* должна существовать документация для пользователя, которая научит его пользоваться программным обеспечением;
* должна существовать возможность экспорта готовых билетов в электронный формат или отправка на печать;
* должен быть реализован интерфейс в консоли, реализованный на вводе данных пользователем;
* сохранение всех данных пользователей, тестов и рейтинга в файлы на жёстком диске

Реализация данных нефункциональных требований позволит достичь при создании программного обеспечения «Конструктор экзаменационных билетов» удобного взаимодействия пользователя с программной, а также простое обучение пользователя, будет обеспечена удобная реализация хранения, а также безопасность всех важных файлов посредством шифрования.

2. Конструирование программы

2.1 Описание модулей программы

При проектировании программного обеспечения конструктор экзаменационных билетов была выбрана модульная архитектура проекта, которая позволит удобно организовать все необходимые функции и классы, а также позволит в перспективе реализовать поддержку, что позволит разрабатывать приложение в команде.

Все основные модули программного обеспечения конструктор экзаменационных билетов представлены на таблице 2.1.

Таблица 2.1 – Описание основных модулей программного средства конструктор экзаменационных билетов

|  |  |  |
| --- | --- | --- |
| Название модуля | Описание | Функционал |
| Модуль регистрации | Обеспечивает процесс регистрации и авторизации пользователей. | * регистрация новых читателей; * авторизация уже зарегистрированных пользователей; * авторизация как гость. |
| Модуль управления тестами | Отвечает за управления тестами. | * создание тестов; * редактирование тестов; * поиск тестов. |
| Модуль работы с билетами | Отвечает за прохождение тестов и выставления оценки за прохождение. | * Прохождение теста; * Просмотр теста; * Выставление оценки. |
| Модуль экспорта | Отвечает за отправление билетов на печать. | * Печать билетов; * Сохранение в электронном формате. |
| Модуль администрирования | Обеспечивает процесс управления пользователями. | * Управление пользователями. |
| Модуль для работы с файлами | Обеспечивает взаимодействие программы с файлами | * Запись в файлы; * Чтение файлов. |

2.2 Выбор способа организации данных

При разработке программного обеспечения «Конструктов экзаменационных билетов» для удобного хранения и быстрого доступа была выбрана файловая структура хранения данных, где некоторые важные поля (например пароль пользователя) зашифрованы обеспечения безопасности пользователей.

Для реализации такого способа хранения данных было разработана следующая структура файлов:

* файл всех пользователей, хранящий ID, имя, логин, пароль, уровень доступа и количество созданных тестов;
* Файл со всеми тестами с пометкой «публичный», такие тесты могут проходить незарегистрированные пользователи;
* Набор файлов, где каждый файл имеет в названии ID пользователя (для удобного поиска). Каждый такой файл хранит данные всех созданных этим пользователем тестов;
* Файл пользовательской документации, для удобного представления документации на выводе и упрощенного редактирования был создан этот файл;
* Файл рейтинга, это файл, хранящий в себя данные каждого пройденного теста, количество прохождений, а также имя пользователя, прошедшего тест и результат прохождения.

Для более удобной для разработчика структуры, файлы хранящие данные тестов находятся в отдельно созданной папке (кроме файла рейтингов), файлы, которые хранят информацию про пользователей также находятся в отдельной папке. Для файла документации была также выделена отдельная папка.

Файлу теста соответствует класс Test (представлен на рисунке 2.1)

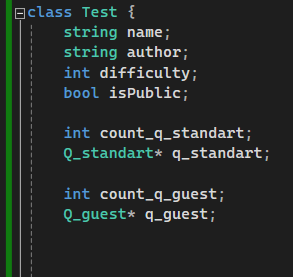


Рисунок 2.1 – реализация класса Test.

Реализация классов Q\_standart и Q\_guest показаны на рисунках 2.2 и 2.3.

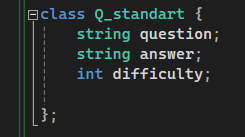


Рисунок 2.2 – реализация класса Q\_standart.

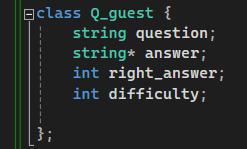


Рисунок 2.3 – реализация класса Q\_guest.

Файлу хранящему данные регистрации соответствует файл users. Этому файлу соответствует класс User, представленный на рисунке 2.4.

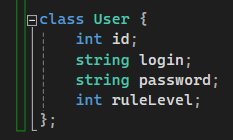


Рисунок 2.4 – реализация класса User.

Файлу хранящему все данные пользователей соответствует файл usersdata. Этому файлу соответствует класс UserData, представленный на рисунке 2.5.

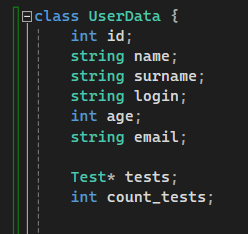


Рисунок 2.5 – реализация класса UserData.

Файлу хранящему все билеты соответствует файл [Имя теста]bilets. Этому файлу соответствует класс BiletsContainer, представленный на рисунке 2.6.

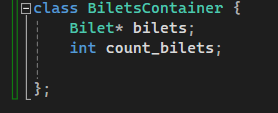


Рисунок 2.6 – реализация класса BiletsContainer.

Внутри класса контейнера билетов хранятся все билеты, им соответствует класс Bilet, представленный на рисунке 2.7.

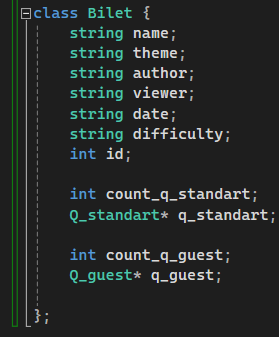


Рисунок 2.7 – реализация класса Bilets.

2.3 Разработка перечня пользовательских функций программы

Необходимый перечень функций, которые будут задействованы в реализации консольной программы на C++ конструктор экзаменационных билетов представлен в таблице 2.2.

Таблица 2.2 – Перечень основных пользовательских функций

|  |  |  |  |
| --- | --- | --- | --- |
| №п/п | Функция | Аргументы | Описание |
| 1 | GetChoice() | Ничего не принимает. | Возвращает тест, который пользователь выбрал из перечня. |
| 2 | CreateExamBilets() | * Int num\_questions (количество вопросов в билете); * Int num\_bilets (Количество билетов). | Возвращает Сгенерированные экзаменационные билеты. |
| 3 | EditTest() | * Test myTest (ранее созданный тест). | Редактирует выбранный тест и возвращает его. |
| 4 | Register() | * string username (Имя пользователя); * string password (Пароль). | Регистрирует нового пользователя, сохраняя данные в файл. |
| 5 | Login() | * string username (Имя пользователя); * string password (Пароль). | Проверяет введенные данные пользователя и разрешает доступ к определенным функциям программы. |
| 6 | Encrypt() | * -string password (Пароль). | Хранит пароли в зашифрованном виде. |
| 7 | CreateTest() | Ничего не принимает. | Отвечает за создание теста и записи его в файл. |
| 8 | TryTest() | * Test test (ранее созданный тест). | Реализует прохождение теста и возвращает оценку за него. |
| 9 | ViewTest() | * Test test (ранее созданный тест). | Показывает тест с ответами. |
| 10 | EditUser() | * User user (зарегистрированный пользователь). | Отвечает за редактирования данных пользователя и его прав доступа (только для администратора). |
| 11 | EditRating() | * Test test (опубликованный тест). | Отвечает за редактирование рейтинга определенного теста (только для создателя теста). |
| 12 | Sort() | * Test \* tests (массив тестов); * string rule (правило сортировки). | Сортирует сходной массив тестов по определенному переданному правилу (переменной). |