Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

|  |  |
| --- | --- |
| Факультет | информационных технологий и управления |
|  |  |
| Кафедра | экономической информатики |

|  |  |
| --- | --- |
|  | *К защите допустить:* |
|  | Руководитель курсового проекта  ассистент кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Купрейчик |

**Пояснительная записка**

к курсовому проекту

на тему

**«АВТОМАТИЗИРОВАННАЯ СИСТЕМА ТЕСТИРОВАНИЯ СТУДЕНТОВ ПО ЯЗЫКУ C++»**

БГУИР КП 6-05-0611-01 102 ПЗ

|  |  |
| --- | --- |
| Студент | А. И. Цевелюк |
| Руководитель: | А. С. Купрейчик |

Минск 2024

**РЕФЕРАТ**

БГУИР КР 6-05-0611-01 102 ПЗ

ТЕМА: Автоматизированная система тестирования студентов по языку C++ / А.И. Цевелюк. – Минск : БГУИР, 2024, – п.з. – 52 с., рисунков – 36, источников – 15, приложений – 2.

*Ключевые слова:* автоматизация, тестирование, система тестирования.

*Объект исследования:* тестирование студентов по языку C++.

*Предмет исследования:* приложение для тестирования студентов по языку C++.

*Цель* *курсовой работы*: автоматизация тестирования студентов по языку C++. Основной задачей является повышение эффективности сбора и анализа данных путем систематизации процесса выставления рейтинга телешоу на канале.

*Методология проведения работы*: в процессе разработки системы использованы методы анализа, классификации, обобщения данных, функциональный анализ процессов, моделирование системы с помощью UML-диаграмм, разработана программная и графическая составляющие программного средства.

*Результаты работы*: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработаны модели бизнес-процессов предметной области на основе нотации IDEF0; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование программного средства, показавшее его соответствие функциональным требованиям, поставленным в задании на разработку. Программный продукт разработан на языке *C++* с применением *MS Visual Studio 2022*.

*Область применения результатов*: разработанная система может быть применена в таких сферах, как частные или государственные УО. Она способствует эффективному управлению информации о тестах, качественной оценке знаний студентов, планированию и принятию качественных решений в сфере образования. Разработанное программное средство полностью отвечает всем функциональным требованиям.

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc184634373)

[1 Анализ и моделирование системы тестирования студентов по  
языку C++ 8](#_Toc184634374)

[1.1 Описание системы тестирования студентов по языку C++ 8](#_Toc184634375)

[1.2 Построение функциональной модели системы тестирования студентов по языку C++ 10](#_Toc184634376)

[2 Проектирование и разработка автоматизированной системы тестирования студентов по языков C++ 18](#_Toc184634377)

[2.1 Информационная модель системы и ее описание 18](#_Toc184634378)

[2.2 Модели представления системы и их описание 21](#_Toc184634379)

[2.2.1 Диаграмма вариантов использования 21](#_Toc184634380)

[2.2.2 Диаграмма последовательностей 23](#_Toc184634381)

[2.2.3 Диаграмма классов 25](#_Toc184634382)

[2.2.4 Диаграмма состояний 27](#_Toc184634383)

[2.3 Описание алгоритмов, реализующих бизнес-логику системы 29](#_Toc184634384)

[2.4 Описание созданных программных конструкций 32](#_Toc184634385)

[3 Описание алгоритма запуска приложения, его использования, результаты работы, тестирования обработки ошибок 37](#_Toc184634386)

[3.1 Алгоритм запуска приложения 37](#_Toc184634387)

[3.2 Руководство пользователя 37](#_Toc184634388)

[3.3 Тестирование работы приложения 42](#_Toc184634389)

[Заключение 45](#_Toc184634390)

[Список использованных источников 46](#_Toc184634391)

[Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» 47](#_Toc184634392)

[Приложение Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику 48](#_Toc184634393)

# ВВЕДЕНИЕ

Автоматизация процессов тестирования студентов по программированию является важной задачей для учебных заведений, преподавателей и студентов [1]. Благодаря внедрению таких систем можно значительно упростить процесс оценки знаний, повысить объективность и сократить время на проверку результатов. Это особенно актуально в условиях дистанционного обучения, когда необходимость в автоматизированных средствах контроля знаний возрастает. Такие системы позволяют оперативно получать обратную связь о степени усвоения материала, а также выявлять пробелы в знаниях студентов, что способствует улучшению учебного процесса [2].

Автоматизированные системы тестирования также обеспечивают возможность многократного проведения тестов, что позволяет преподавателям гибко адаптировать контроль знаний к различным условиям обучения. Кроме того, данные системы дают возможность создания базы тестовых заданий, которая может использоваться для различных курсов и предметов. Это снижает нагрузку на преподавателей, обеспечивая единый стандарт оценки знаний, и позволяет поддерживать высокий уровень объективности и прозрачности процесса тестирования. Для студентов автоматизация позволяет улучшить навыки самопроверки и самостоятельной подготовки к экзаменам, поскольку они могут получать немедленную обратную связь по результатам тестирования [3].

Разработка системы тестирования студентов по программированию на языке C++ с использованием современных технологий и подходов объектно-ориентированного программирования является важной задачей в условиях цифровой трансформации образования. Это позволяет интегрировать различные методы работы с данными, шифрование, обеспечение безопасности личной информации и удобство работы с системой для пользователей [13].

Цель данной курсовой работы — автоматизация тестирования студентов по языку программирования C++. Для этого необходимо выполнить следующие задачи:

– изучить и проанализировать бизнес-процесс системы тестирования студентов по языку C++, визуализировать функциональную модель системы при помощи стандарта IDEF0;

– спроектировать архитектуру системы с использованием стандарта моделирования UML и блок-схем, разработать автоматизированную систему тестирования студентов по языку C++ в соответствии с спроектированной архитектурой;

– отладить и протестировать систему для обеспечения корректной работы, описать алгоритм запуска разработанного программного обеспечения, написать легкое в освоении конечным пользователем руководство.

Объектом исследования является автоматизированная система тестирования студентов. Предметом исследования – процесс разработки и внедрения программного обеспечения для автоматизированного контроля знаний.

Теоретической и методологической основой данной работы стали научные публикации и учебные пособия, посвященные методам автоматизации учебных процессов, а также современные стандарты разработки программных систем.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет 89,23%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников». Скриншот приведен в приложении А (рисунок А.1).

# 1 АНАЛИЗ И МОДЕЛИРОВАНИЕ СИСТЕМЫ ТЕСТИРОВАНИЯ СТУДЕНТОВ ПО ЯЗЫКУ C++

## 1.1 Описание системы тестирования студентов по языку C++

Система автоматизированного тестирования студентов по языку программирования C++ представляет собой специализированное программное обеспечение, предназначенное для организации и проведения контроля знаний учащихся. Предметная область охватывает широкий спектр задач, связанных с оценкой уровня знаний студентов, автоматизацией процессов создания и редактирования тестов, а также хранением и обработкой результатов. В образовательных учреждениях такие системы играют важную роль, поскольку позволяют сократить временные затраты преподавателей на проверку тестов и обеспечивают объективную оценку знаний студентов. Кроме того, автоматизированное тестирование предоставляет возможность регулярной самопроверки для учащихся, что способствует лучшему усвоению материала.

В последние годы системы автоматизированного тестирования стали неотъемлемой частью образовательного процесса, особенно в условиях массового дистанционного обучения. Такие системы позволяют проводить тестирование в любое время, независимо от физического присутствия студентов и преподавателей, что делает образовательный процесс более гибким и доступным. Кроме того, современные системы тестирования предоставляют широкий набор функциональных возможностей, включая авторизацию, защиту данных, генерацию отчетов и управление базами тестов, что делает их важным инструментом для образовательных учреждений.

На сегодняшний день существует множество программных аналогов систем тестирования, которые активно используются в образовательных учреждениях по всему миру. К ним относятся такие системы, как Moodle [4], Blackboard [5], Google Classroom и другие. Эти системы предоставляют преподавателям и учащимся инструменты для создания тестов, их прохождения и анализа результатов. Они поддерживают различные форматы вопросов (например, с выбором одного или нескольких правильных ответов, с вводом текста, с использованием математических формул и т. д.), что позволяет адаптировать тестирование к различным предметам и дисциплинам.

Однако большинство существующих систем, таких как Moodle и Blackboard, являются комплексными платформами для управления учебным процессом и включают в себя не только функции тестирования, но и другие инструменты для взаимодействия преподавателей и студентов. Они часто требуют подключения к сети Интернет, работы на серверах и поддержки большого количества пользователей. В отличие от них, предлагаемая система тестирования по языку C++ ориентирована на выполнение одной конкретной задачи – автоматизацию проверки знаний студентов в области программирования [6]. Это позволяет сосредоточиться на разработке специализированных алгоритмов для проверки знаний, таких как обработка кода, анализ ошибок и выполнение программных заданий.

Основная задача системы автоматизированного тестирования студентов заключается в создании функционала для проведения тестов, хранения результатов и обеспечения их объективности. Для решения этой задачи применяются различные методы программирования, включая объектно-ориентированное программирование (ООП), работу с файлами, обработку ошибок и шифрование данных. Одним из ключевых элементов системы является реализация авторизации и разграничения прав доступа пользователей. Преподаватели получают возможность создавать и редактировать тесты, а студенты — проходить их и просматривать свои результаты.

Одним из важнейших методов, применяемых в системе, является использование контейнеров библиотеки стандартных шаблонов C++ (STL). Эти контейнеры обеспечивают эффективное хранение и обработку данных о студентах, тестах и результатах тестирования. Например, для хранения вопросов и ответов в тестах может использоваться структура данных на основе std::map, что позволяет быстро находить ответы на вопросы по ключу (вопросу). Кроме того, использование контейнера std::vector для хранения списка тестов и результатов позволяет легко выполнять такие операции, как сортировка и фильтрация данных.

Для защиты конфиденциальных данных, таких как пароли пользователей, применяется метод шифрования с использованием простого криптографического алгоритма на основе операции XOR, модифицированной с учетом длины пароля. Этот метод обеспечивает базовую защиту данных, что особенно важно в условиях хранения персональной информации студентов.

Автоматизация процесса проверки знаний предполагает также создание алгоритмов для оценки правильности ответов студентов. В зависимости от типа вопросов (например, с выбором правильного ответа или с вводом текста) система может проверять соответствие введенного ответа эталонному значению. Для этого реализуется логика сопоставления ответов с базой данных тестов, что позволяет автоматически начислять баллы и генерировать итоговый результат.

Существующие программные решения, такие как Moodle и Blackboard, предоставляют широкий набор возможностей, однако они имеют ряд ограничений для конкретных задач тестирования студентов по языку C++. Во-первых, данные системы часто требуют сложной настройки и администрирования, что может затруднять их использование в небольших учебных заведениях или при ограниченных ресурсах. Во-вторых, большинство аналогичных систем ориентированы на общий образовательный процесс и не предоставляют специализированных инструментов для проверки знаний в области программирования, таких как компиляция кода и анализ ошибок.

Предлагаемая система, в отличие от крупных платформ управления обучением, не требует подключения к Интернету и может быть запущена на любом компьютере с операционной системой Windows. Это обеспечивает высокую мобильность и независимость системы от внешних сервисов. Она также адаптирована для работы в среде программирования, что позволяет проверять не только теоретические знания, но и практические навыки студентов, такие как написание кода и решение задач на языке C++.

Система тестирования студентов по языку C++ решает важную задачу автоматизации процесса проверки знаний. Применение современных методов программирования, таких как объектно-ориентированное программирование, работа с файлами и шифрование данных, позволяет создать эффективное и надежное программное решение. В отличие от существующих аналогов, данная система ориентирована на решение конкретных задач тестирования по программированию, что делает ее более удобной и функциональной для преподавателей и студентов.

## 1.2 Построение функциональной модели системы тестирования студентов по языку C++

IDEF0 (Integration Definition for Function Modeling) – это методология и язык моделирования, используемые для анализа и проектирования бизнес-процессов [7]. Она позволяет описывать функции, потоки данных и контроль в бизнес-процессах и представляет их в виде блок-схем.

Использование IDEF0 позволяет более эффективно проектировать бизнес-процессы, повышать качество их выполнения и уменьшать затраты на их выполнение. Она является надежным инструментом для анализа и проектирования систем, включая систему по тестированию студентов по языку C++.

Рассмотрим информационную систему, выполненную с помощью средств моделирования функций IDEF0. Для начала необходимо сделать контекстную модель информационной системы. Контекстная диаграмма – самая верхняя диаграмма, на которой объект моделирования представлен единственным блоком с граничными стрелками. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой.

На рисунке 1.1 описан основной блок «Протестировать студента». Входящие стрелки – «Данные о студенте», «Данные о вопросах в тесте». Это то, что необходимо иметь для начала работы. Стрелки управления – «Правила тестирования в УО», «Законодательные и нормативные акты». В роли механизмов выступают преподаватель и программное обеспечение. После завершения процесса мы получаем «Результаты тестирования».

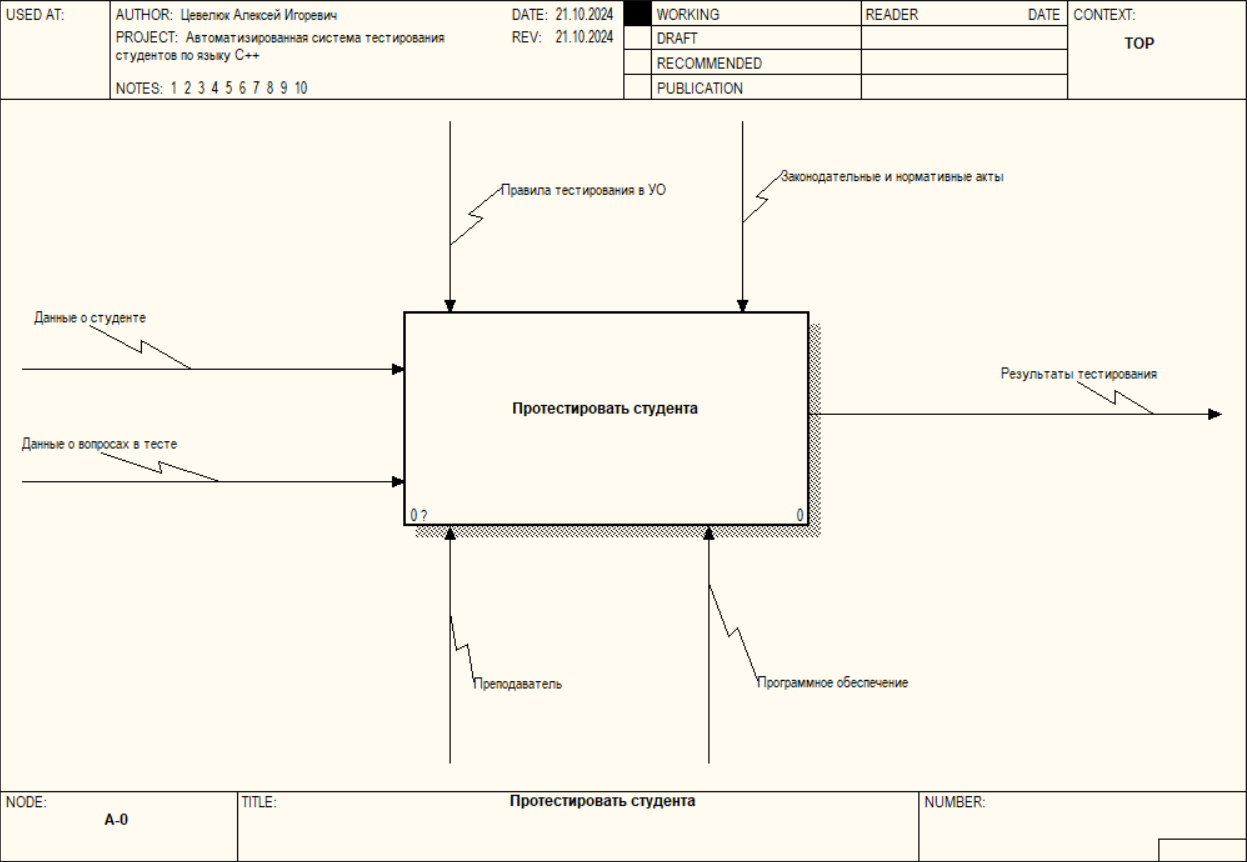


Рисунок 1.1 – Контекстная диаграмма модели А-0

На рисунке 1.2 представлена декомпозиция процесса «Протестировать студента». В данном случае мы получили диаграмму, состоящую из четырех процессов:

1. Авторизовать студента – процесс авторизации студента, который будет проходить тестирование.
2. Составить тест – процесс составления теста, который будет проходить студент.
3. Провести тестирование – процесс прохождения тестирования авторизованным студентом.
4. Подвести результаты тестирования ­­­­– процесс, в результате которого студент и преподаватель получает количество правильных и неправильных ответов, а также отметку за прохождение теста.

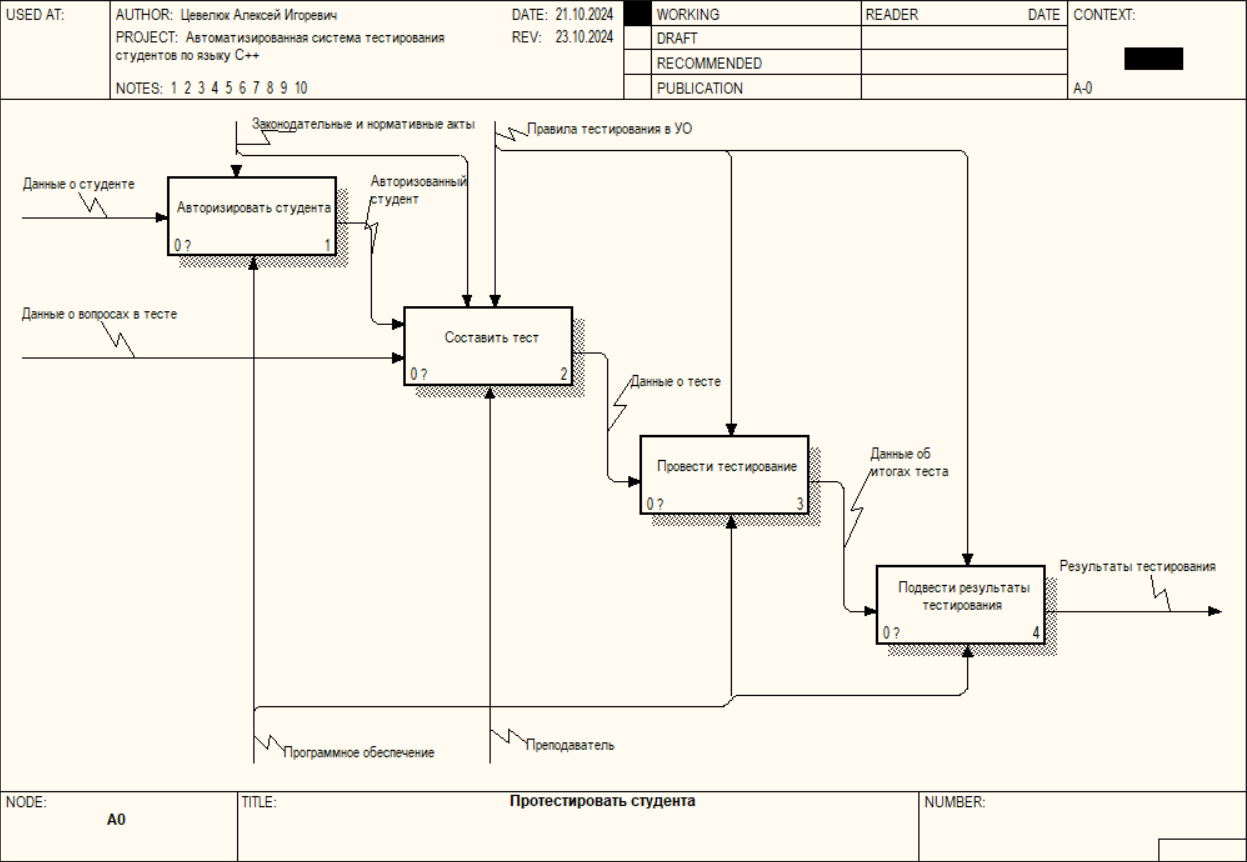


Рис. 1.2 – Декомпозиция блока «Протестировать студента»

На рисунке 1.3 предоставлена декомпозиция процесса «Авторизовать студента». Для этого нужно получить логин, получить пароль, проверить введенные данные и, если логин и пароль подходящие, подтвердить авторизацию.

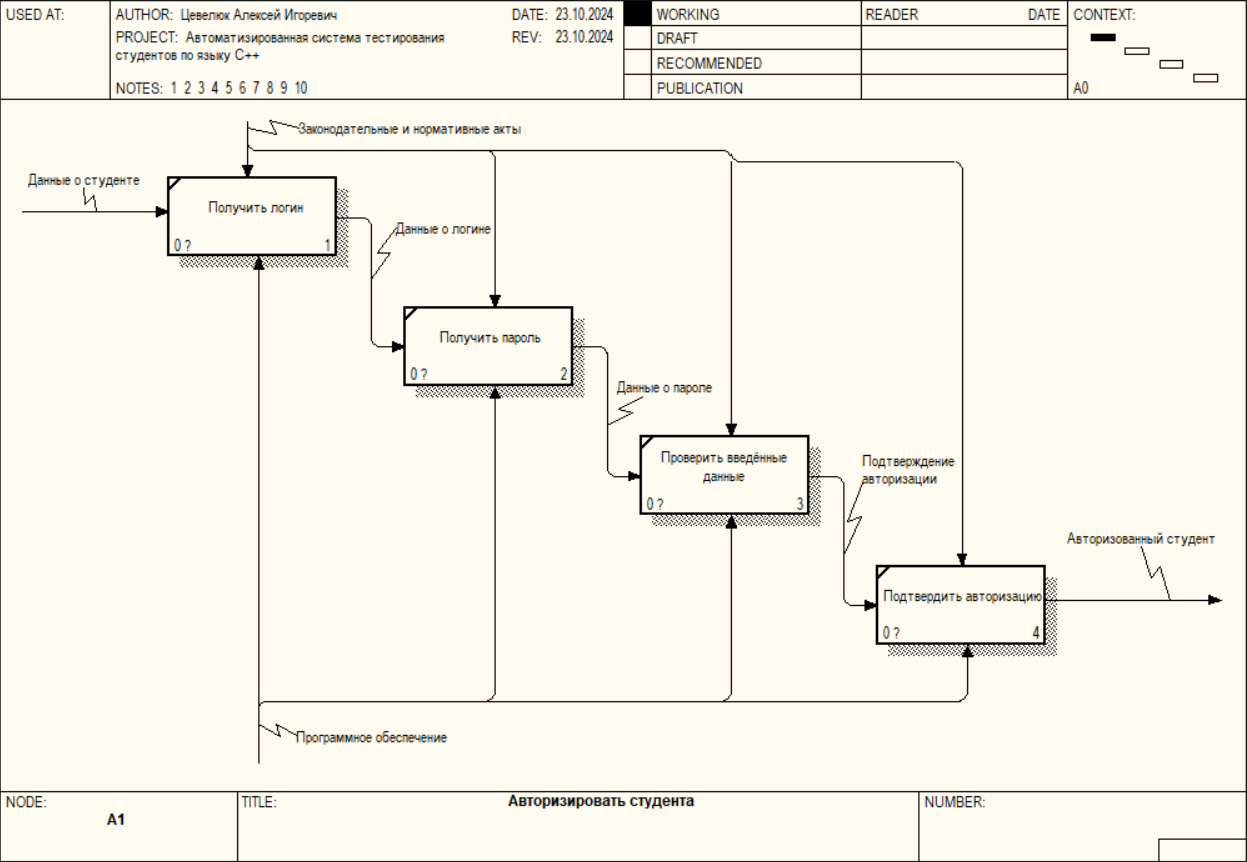


Рис. 1.3 – Декомпозиция блока «Авторизовать студента»

На рисунке 1.4 предоставлена декомпозиция процесса «Составить тест». Этот блок включает в себя составление вопросов и ответов к ним, а затем соединение всего воедино.

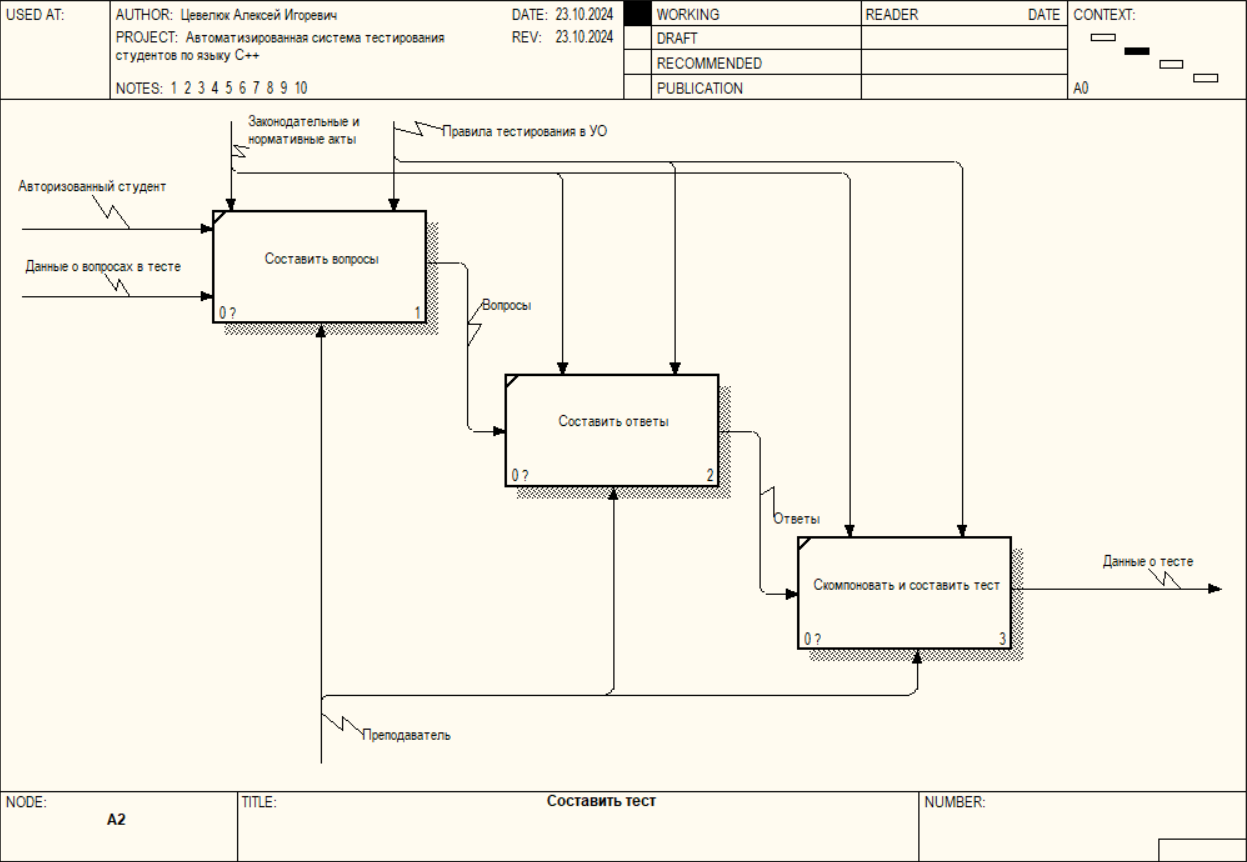


Рис. 1.4 – Декомпозиция блока «Составить тест»

На рисунке 1.5 предоставлена декомпозиция процесса «Провести тестирование». Тестирование проводится путем вывода на экран вопросов и получение ответов на них от студента.

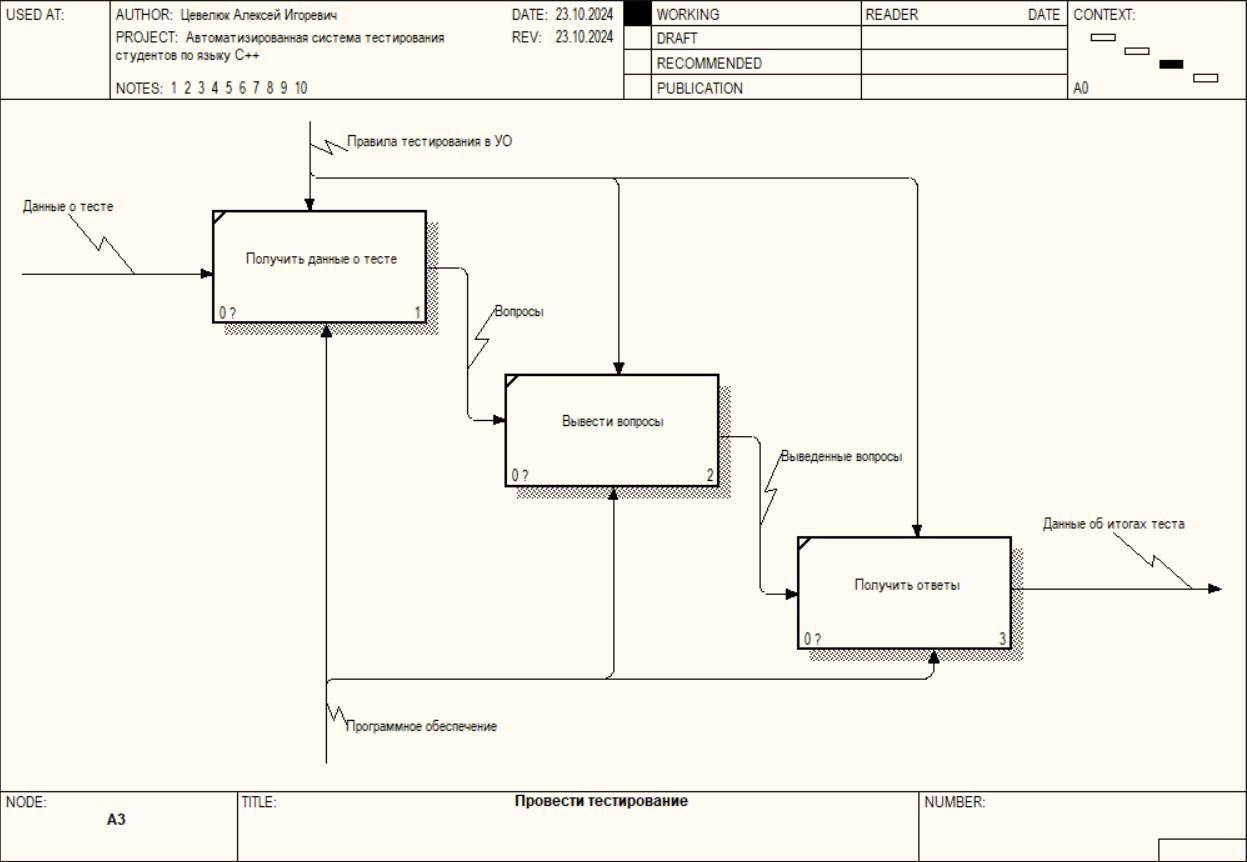


Рис. 1.5 – Декомпозиция блока «Провести тестирование»

На рисунке 1.6 предоставлена декомпозиция процесса «Подвести результаты тестирования». Для этого необходимо посчитать количество правильных и неправильных ответов, а затем сформировать отметку за тест.

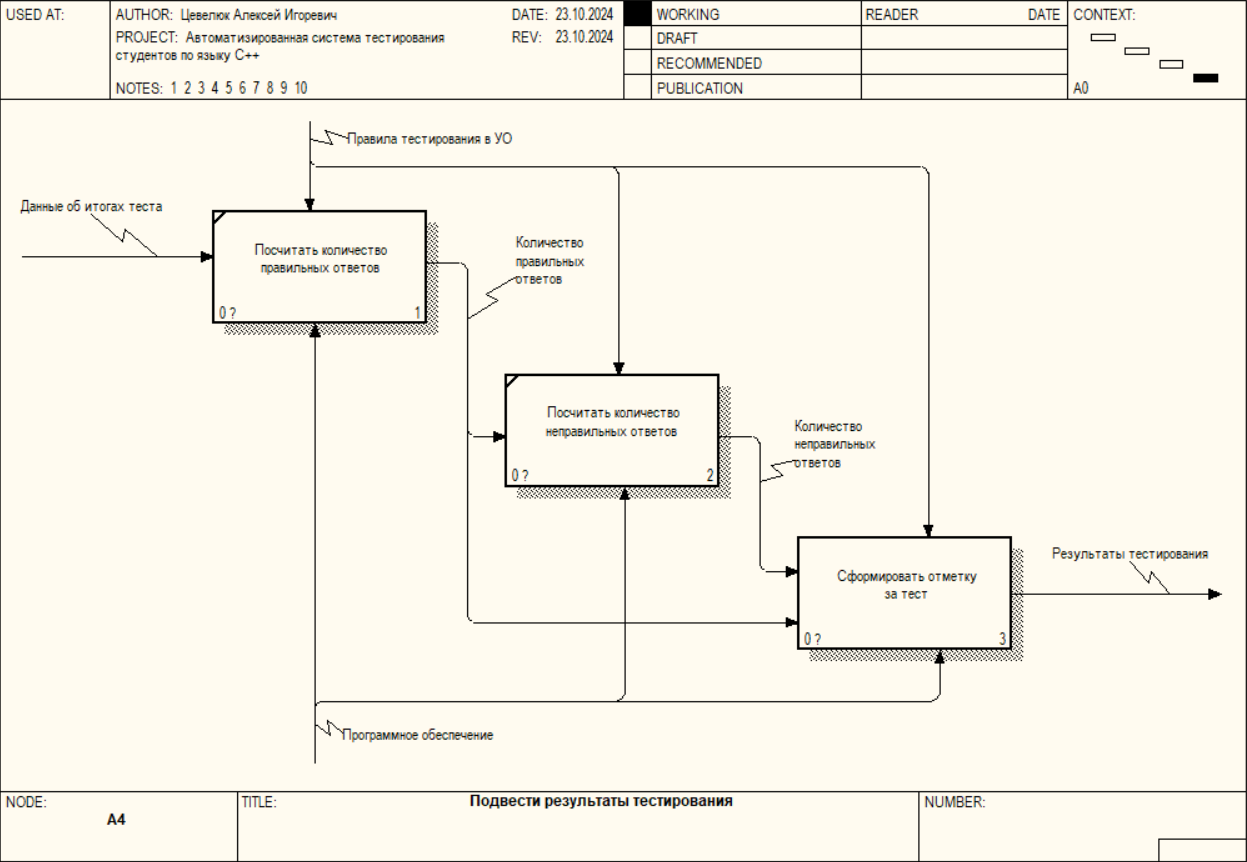


Рис. 1.6 – Декомпозиция блока «Подвести результаты тестирования»

На рисунке 1.7 предоставлена декомпозиция процесса «Сформировать отметку за тест». Чтобы выставить ее студенту, необходимо получить данные о количестве правильных и неправильных ответов, после чего рассчитать отметку по специальной формуле.

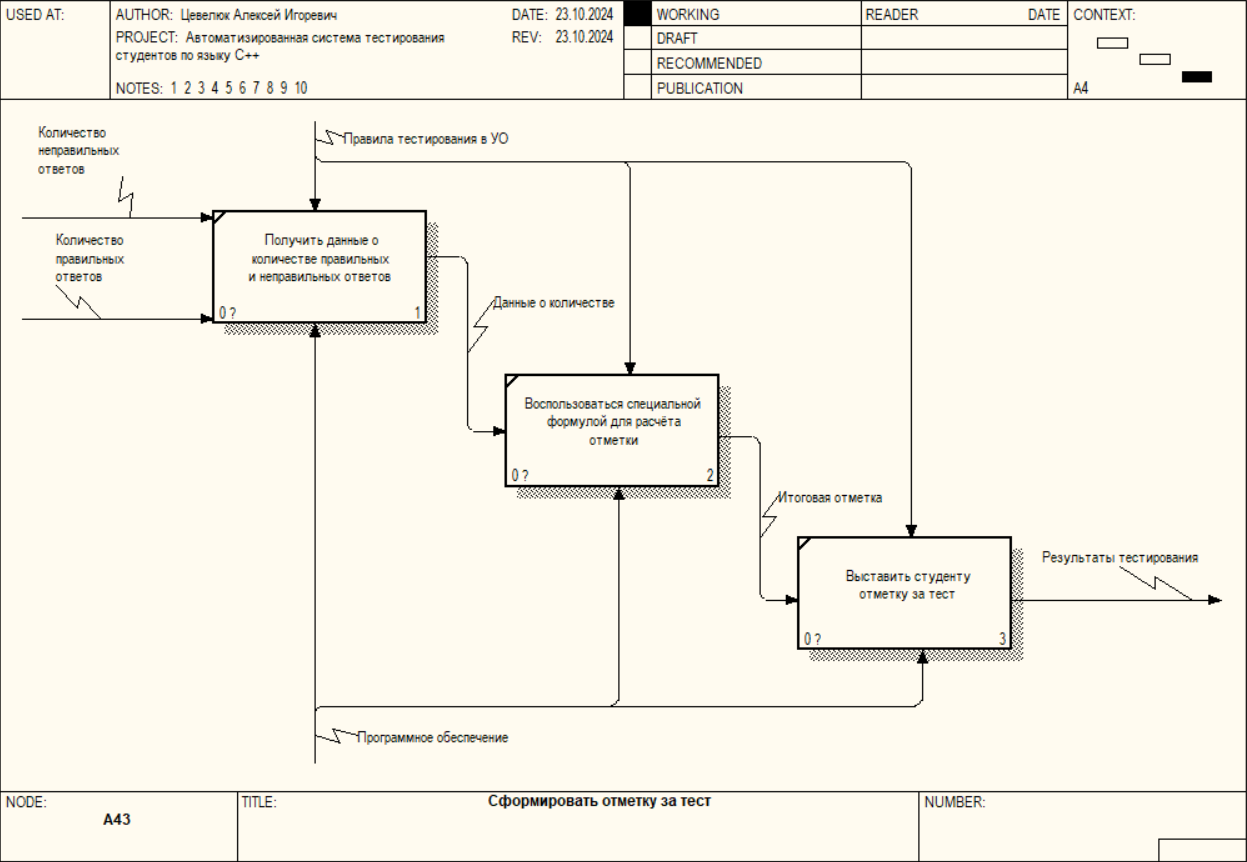


Рис. 1.7 – Декомпозиция блока «Сформировать отметку за тест»

Таким образом, мы визуализировали организацию и задачи нашей системы, используя язык IDEF0.

Анализ и моделирование бизнес-процессов, связанных с тестированием студентов по C++, с использованием диаграмм IDEF0, позволяет лучше понять структуру системы, визуализировать модели объектов и процессов, определить потоки информации и выявить возможные узкие места и проблемы в процессе. Использование такого подхода помогает разработчикам создать эффективную и удобную систему тестирования студентов по языку C++, которая удовлетворит потребности пользователей и бизнес-процессов.

# 2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ТЕСТИРОВАНИЯ СТУДЕНТОВ ПО ЯЗЫКОВ C++

В рамках данной курсовой работы разрабатывается автоматизированная система тестирования студентов, ориентированная на проверку знаний по языку программирования C++. Данная система представляет собой программное обеспечение, в основе которого лежат принципы объектно-ориентированного программирования, что позволяет обеспечить структурированный и удобный подход к разработке.

Основу системы составляют данные о пользователях и тестах, которые будут представлены в виде объектов с четко определенными атрибутами и методами. Это обеспечивает гибкость в обработке информации и упрощает взаимодействие между компонентами программы. Реализация системы предполагает создание набора классов, моделирующих ключевые элементы системы: учетные записи пользователей, тесты, вопросы и ответы.

Использование объектно-ориентированного подхода позволяет не только систематизировать процесс разработки, но и заложить возможности для дальнейшего масштабирования и улучшения программы. Такой подход также способствует повышению читаемости кода и упрощению его сопровождения.

Проект ориентирован на применение C++ как основного языка разработки, что обусловлено его широкими возможностями для реализации объектно-ориентированных концепций, таких как наследование, полиморфизм и инкапсуляция.

# 2.1 Информационная модель системы и ее описание

Разработка автоматизированной системы тестирования студентов требует тщательной проработки информационной модели, которая задает структуру и взаимодействие между элементами системы. В данной части описывается структура хранимых данных, формат файлов, используемых в системе, и их взаимосвязь.

Для реализации автоматизированной системы тестирования студентов используется файловая система, состоящая из трех текстовых файлов: results.txt, tests.txt и users.txt. Каждый из них имеет свое назначение, формат данных и правила обработки.

results.txt. Этот файл хранит результаты тестирования студентов. Каждая строка файла соответствует одному прохождению теста и содержит следующую информацию:

– логин студента;

– название теста;

– количество правильных ответов.

Формат записи:

логин\_студента;название\_теста;количество\_правильных\_ответов

Пример:

ivanov;C++\_Basics;7

petrov;OOP\_Advanced;10

sidorov;C++\_Basics;5

tests.txt. Этот файл содержит описание тестов. Каждая тестовая запись состоит из нескольких частей:

– название теста;

– вопросов;

– список вопросов, где каждый вопрос описывается текстом, вариантом ответов и номером правильного ответа.

Формат записи:

название\_теста

количество\_вопросов

вопрос;ответ1,ответ2,ответ3,ответ4;номер\_правильного\_ответа

…

Пример:

C++\_Basics

2

What is a pointer?;A variable,A function,A memory address,None of the above;2

Which of these is a loop construct in C++?;if,while,int,double;1

users.txt Данный файл содержит информацию о зарегистрированных пользователях системы. Запись включает:

– логин;

– пароль в зашифрованном виде;

– флаг администратора (1 ­– администратор, 0 – обычный пользователь).

Формат записи:

логин пароль\_в\_зашифрованном\_виде флаг\_админа

Пример:

admin f24a72cb1df1 1

student1 a81bce42a12d 0

student2 b5f34e912c34 0

Файлы тесно взаимосвязаны через логику программы:

– users.txt используется для аутентификации пользователей и проверки их роли (администратор или студент).

– tests.txt предоставляет вопросы и ответы для прохождения тестов.

– results.txt хранит результаты, которые анализируются и используются для дальнейшего отображения прогресса.

IDEF1X-схема структуры файлов и их взаимодействия представлена ниже:

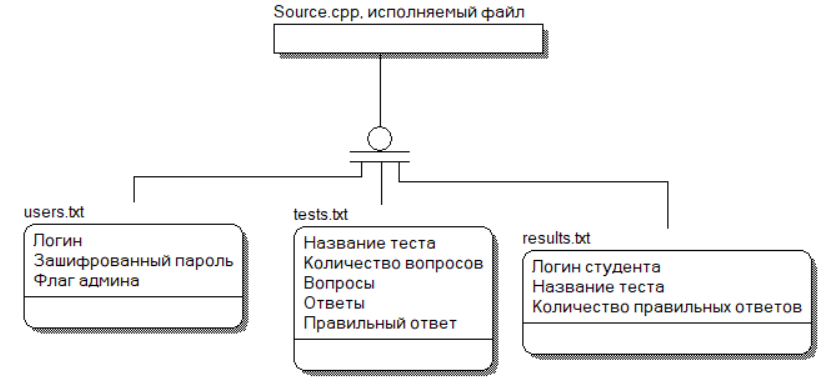


Рис. 2.1 – IDEF1X-схема структуры файлов и их взаимодействия

Взаимодействие файлов можно описать следующими процессами:

1 Аутентификация. При входе в систему программа обращается к users.txt, чтобы проверить логин и пароль пользователя. Флаг администратора определяет доступ к функциям редактирования тестов.

2 Прохождение теста. При старте тестирования программа считывает соответствующий тест из tests.txt. Студент отвечает на вопросы, а результаты записываются в results.txt.

3 Анализ результатов. Для отображения прогресса и итогов тестирования используется results.txt, данные из которого сопоставляются с информацией из users.txt.

Преимущества разработанной файловой структуры:

1 Простота реализации: текстовые файлы не требуют сложной настройки.

2 Удобство переноса данных: файлы легко переносятся и читаются вручную.

3 Минимальные системные требования: текстовые файлы подходят для программ, выполняемых на компьютерах с ограниченными ресурсами.

Таким образом, информационная модель системы полностью отвечает требованиям функциональности, обеспечивая простоту и эффективность.

# 2.2 Модели представления системы и их описание

В данном подразделе будет продемонстрировано моделирование автоматизированной системы тестирования студентов по языку C++ с использованием стандарта UML (Unified Modeling Language). Этот стандарт применяет графические обозначения для создания абстрактной модели системы и предназначен для определения, визуализации, проектирования и документирования программных систем.

UML предоставляет возможность описывать систему с разных точек зрения, таких как:

– статическая структура системы (классы, объекты, их атрибуты и отношения);

– динамическое поведение, включающее взаимодействие между компонентами системы, последовательность выполнения операций и управление потоками данных.

Использование UML позволяет не только детализировать архитектуру программного обеспечения, но и сделать проект более понятным для разработчиков, тестировщиков и других участников процесса. Это способствует формализации требований, улучшает согласованность между всеми этапами разработки и обеспечивает полноценную документацию проекта.

# 2.2.1 Диаграмма вариантов использования

Диаграмма вариантов использования – это один из ключевых инструментов UML, который позволяет визуализировать взаимодействие между пользователями (акторами) и системой. Она предоставляет наглядное представление о том, какие функции доступны различным ролям, а также помогает разработчикам понять, как пользователи будут работать с системой.

В рамках данной автоматизированной системы тестирования студентов по языку C++ диаграмма вариантов использования помогает проанализировать задачи, которые пользователи должны выполнять. Это критически важно для проектирования удобного интерфейса и обеспечения необходимой функциональности.

Преимущества диаграммы вариантов использования:

1 Простота и ясность. Диаграмма позволяет легко понять, какие задачи решает система, и как она взаимодействует с различными пользователями. Это важно для дальнейшего проектирования интерфейсов и логики.

2 Гибкость в дополнении. Если в будущем потребуется добавить новую функцию (например, анализ статистики успехов студентов), ее легко интегрировать в текущую модель.

3 Понимание для всех участников разработки. Диаграмма доступна для понимания не только программистам, но и другим участникам проекта, включая преподавателей, заказчиков или даже студентов, тестирующих систему.

Обоснование выбора вариантов использования:

1 Роли в системе. В системе выделяются две основные роли: Администратор и Студент. Администратор отвечает за управление тестами и контроль за результатами студентов. Это позволяет ему оперативно реагировать на изменения и поддерживать актуальность тестов. Студент пользуется системой для прохождения тестов, анализа своих результатов и работы с каталогом доступных заданий.

2 Уникальные варианты использования. Каждый вариант использования был тщательно выбран на основе ключевых сценариев работы системы. Например, создание тестов и прохождение тестов — это основные процессы для администратора и студента соответственно. Возможность сортировки, фильтрации и поиска тестов упрощает навигацию, а просмотр результатов делает взаимодействие с системой более полезным.

3 Взаимодействие с файлами Использование текстовых файлов в качестве хранилищ данных (тестов, результатов, пользователей) также учитывается в диаграмме. Например, при создании тестов или записи результатов прохождения задействуются соответствующие файлы, что отображает связь между функциональностью и структурой данных системы.

В результате была разработана диаграмма вариантов использования, показанная ниже:

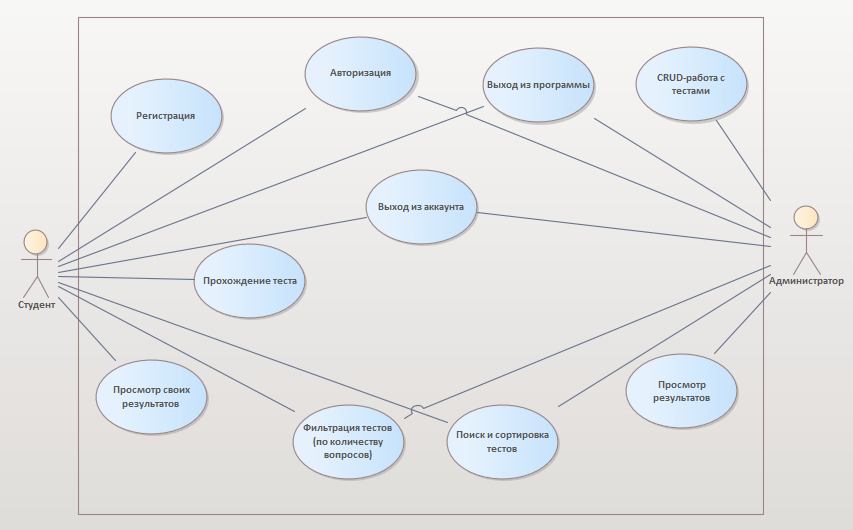


Рис. 2.2 – UML-диаграмма вариантов использования

Описание вариантов использования исключительно для студента:

1 Регистрация. Студенты должны иметь возможность зарегистрироваться в системе, используя логин и пароль.

2 Прохождение теста. Является основным функционалом для студента, позволяет ему пройти тест, разработанный заранее.

3 Просмотр своих результатов. Отображает результаты по всем тестам, что когда-либо проходил студент.

Описание вариантов использования исключительно для администратора:

1 CRUD-работа с тестами. Администратор должен иметь возможность совершать CRUD-работу (Create, Read, Update, Delete) над тестами, чтобы грамотно управлять системой.

2 Просмотр результатов. Администратор может просмотреть результаты всех студентов по всем тестам, что когда-либо проходил студент.

Описание вариантов использования и для студента, и для администратора:

1 Авторизация. Вне зависимости от роли, система потребует перед использованием ввести свой логин и пароль, чтобы авторизироваться в ней.

2 Поиск и сортировка тестов. Любой пользователь может совершить поиск или сортировку тестов по заданным параметрам.

3 Фильтрация тестов. Любой пользователь может произвести фильтрацию тестов по количеству вопросов.

4 Выход из аккаунта. После завершения работы, пользователь может выйти из своей учетной записи, предоставив возможность войти другому.

5 Выход из программы. После того, как все студенты и администраторы закончили свои сессии в программе, ее можно закрыть безопасным способом, сохранив все изменения в файлах системы.

Таким образом, диаграмма вариантов использования является важным этапом моделирования системы. Она демонстрирует ключевые аспекты ее поведения и взаимодействия с пользователями, создавая основу для детальной проработки остальных моделей. В данном проекте диаграмма способствует структурному подходу к проектированию, что в итоге улучшает ее функциональность, производительность и удобство для пользователей.

# 2.2.2 Диаграмма последовательностей

Диаграмма последовательностей — это диаграмма поведения, которая показывает взаимодействие объектов системы и подчеркивает временную последовательность событий [9]. В рамках данной работы диаграмма последовательностей используется для точного определения логики сценария выполнения ключевых вариантов использования.

Диаграммы последовательностей помогают разработчикам понять, какие объекты участвуют в выполнении сценария, какие сообщения передаются между ними и в каком порядке эти сообщения отправляются. Также отображаются возможные возвращаемые значения, что важно для отладки и проверки работы системы.

Рассмотрим сценарий «Прохождение теста студентом».

На диаграмме последовательностей рассматривается пример сценария, где студент авторизуется в системе, выбирает тест, проходит его и сохраняет результаты. Основными участниками взаимодействия в данном сценарии являются:

– пользователь (актор, представляющий студента);

– система (обеспечивает интерфейс и выполняет основные функции);

– объекты теста и результата (обеспечивают загрузку и сохранение данных).

В результате была разработана диаграмма последовательностей, показанная ниже:

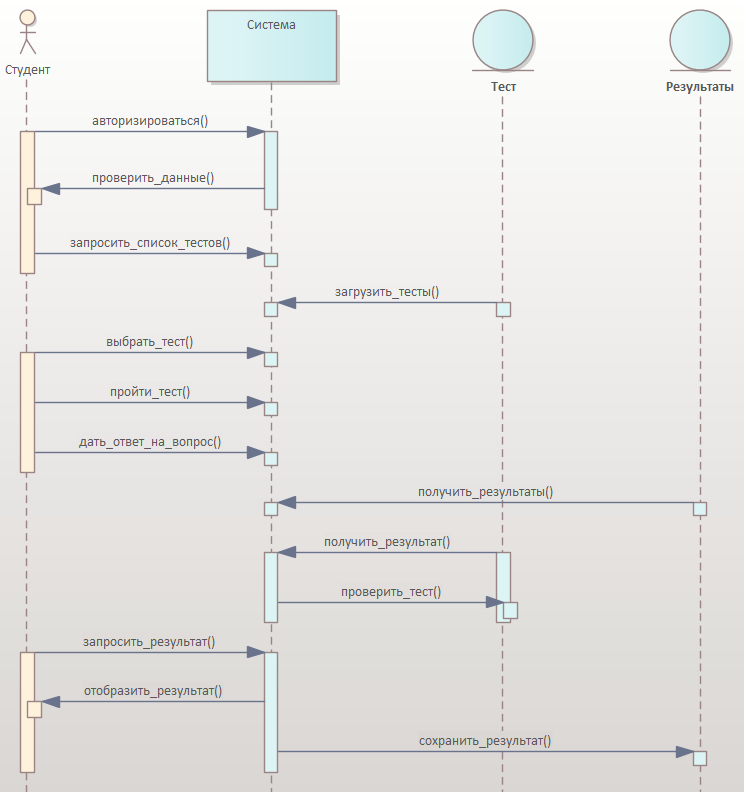


Рис. 2.3 – UML-диаграмма последовательностей

Расшифровка диаграммы:

1 Авторизация. Пользователь вводит свои учетные данные, которые проверяются системой. При успешной авторизации предоставляется доступ к интерфейсу.

2 Выбор теста. Система загружает список доступных тестов и отображает его пользователю. Пользователь выбирает нужный тест.

3 Прохождение теста. После выбора теста пользователь отвечает на предложенные вопросы. Система собирает ответы и подсчитывает итоговый результат.

4 Сохранение результата. Полученный результат сохраняется в соответствующем файле.

5 Отображение результата. Пользователь видит свой результат на экране, который также может быть доступен для дальнейшего просмотра.

Диаграмма последовательностей позволяет детализировать процесс выполнения сценария "Прохождение теста студентом". Она наглядно демонстрирует взаимодействие между пользователем, системой и ее компонентами, а также уточняет порядок обмена данными. Это помогает не только разработчикам понять логику системы, но и выделить возможные узкие места в сценарии, требующие оптимизации.

Использование данной диаграммы на этапе проектирования системы способствует улучшению структуры кода, упрощает тестирование и дальнейшую доработку функционала.

# 2.2.3 Диаграмма классов

Диаграмма классов является одним из ключевых инструментов объектно-ориентированного проектирования и моделирования систем [8]. Она описывает структуру системы, определяя классы, их атрибуты, методы, а также связи между классами, такие как ассоциации, наследование и зависимости.

Диаграмма классов позволяет разработчику:

– увидеть общую архитектуру системы;

– определить основные сущности и их взаимодействие;

– наглядно представить реализацию бизнес-логики;

– выявить потенциальные места улучшений структуры кода.

На представленной диаграмме классов системы автоматизированного тестирования студентов по языку C++ выделены основные классы: question, test, user, result, application, а также интерфейс i\_manageable.

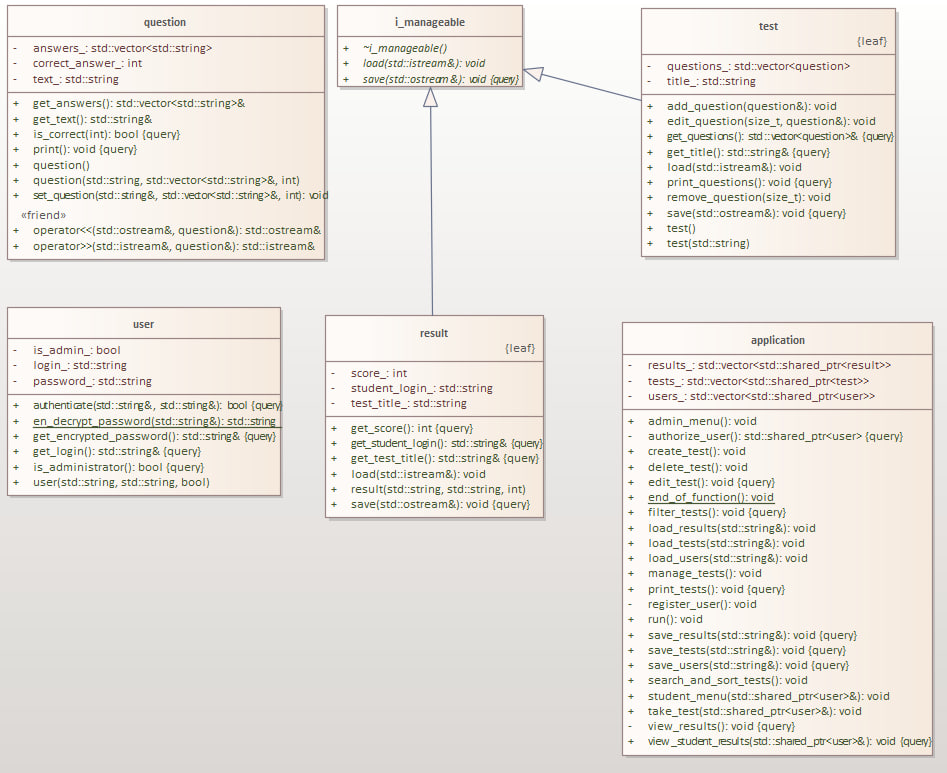


Рис. 2.4 – UML-диаграмма классов

Основные элементы диаграммы:

1 Интерфейс i\_manageable. Этот интерфейс объединяет методы, которые можно реализовать для управления сохранением и загрузкой данных (например, из файлов). Его реализуют классы question, test и result. Это позволяет добиться единого подхода к работе с объектами, минимизируя дублирование кода.

2 Класс question. Назначение: отвечает за хранение информации о вопросах теста (включая текст вопроса, варианты ответов и правильный ответ). Ключевые методы: предоставляют доступ к данным вопроса и проверяют правильность ответа. Причина включения: вопросы являются основой любого теста, и их данные должны быть четко структурированы.

3 Класс test Назначение: управляет коллекцией вопросов, связанными с конкретным тестом. Ключевые методы: позволяют добавлять, редактировать, удалять вопросы, а также загружать и сохранять тесты. Причина включения: тесты являются логическим объединением вопросов, предоставляя функционал для работы с ними.

4 Класс user Назначение: хранит данные о пользователях системы (логин, пароль и права доступа). Ключевые методы: отвечают за авторизацию, а также за шифрование и дешифрование пароля. Причина включения: система должна различать студентов и администраторов, что обеспечивает гибкость и безопасность.

5 Класс result Назначение: хранит результаты прохождения тестов (оценка, логин студента и название теста). Ключевые методы: позволяют получать данные о результатах и сохранять их. Причина включения: сохранение результатов необходимо для анализа успеваемости студентов.

6 Класс application Назначение: объединяет основные функции системы, такие как авторизация, работа с тестами и результатами, взаимодействие с пользователями. Ключевые методы: включают управление тестами, пользователями, а также загрузку/сохранение данных. Причина включения: обеспечивает взаимодействие всех остальных классов и выполняет роль главного контроллера системы.

Связи между классами:

– наследование: классы question, test, result наследуют интерфейс i\_manageable, чтобы унифицировать операции загрузки и сохранения данных;

– ассоциации: класс application содержит коллекции объектов user, test и result. Это обеспечивает управление всеми ключевыми компонентами системы из одного класса;

– композиция: класс test включает в себя коллекцию объектов question, так как вопросы являются неотъемлемой частью тестов.

Диаграмма классов разработанной системы позволяет структурировать и упрощать разработку, так как она наглядно демонстрирует взаимодействие между ключевыми элементами. Выбранные классы и их связи полностью отвечают функциональным требованиям к системе: возможность тестирования студентов, авторизации пользователей, управления тестами и сохранения результатов.

Интерфейсное наследование и ассоциации между классами создают гибкую архитектуру, которая позволяет легко расширять систему и добавлять новый функционал. Это делает проект модульным, поддерживаемым и понятным для других разработчиков.

# 2.2.4 Диаграмма состояний

Диаграмма состояний – это диаграмма поведения, которая моделирует жизненный цикл объекта. Она показывает, в каких состояниях может находиться объект, а также переходы между этими состояниями, вызванные внешними или внутренними событиями.

Объекты системы изменяют свое состояние в зависимости от поступающих событий или выполненных действий. Каждое состояние характеризуется особыми параметрами, а переходы между состояниями указывают на реакцию системы на определенные события. Диаграмма состояний помогает проанализировать и понять динамическую природу объекта в системе, что делает ее незаменимым инструментом при проектировании сложных приложений.

Диаграмма состояний особенно полезна в системах, где важно отслеживать последовательность действий или событий, влияющих на поведение компонентов.

В контексте автоматизированной системы тестирования студентов по языку C++ объектом, для которого логично построить диаграмму состояний, является процесс прохождения теста студентом. На диаграмме отражены ключевые этапы прохождения теста, включая начальное и конечное состояния, а также промежуточные действия.

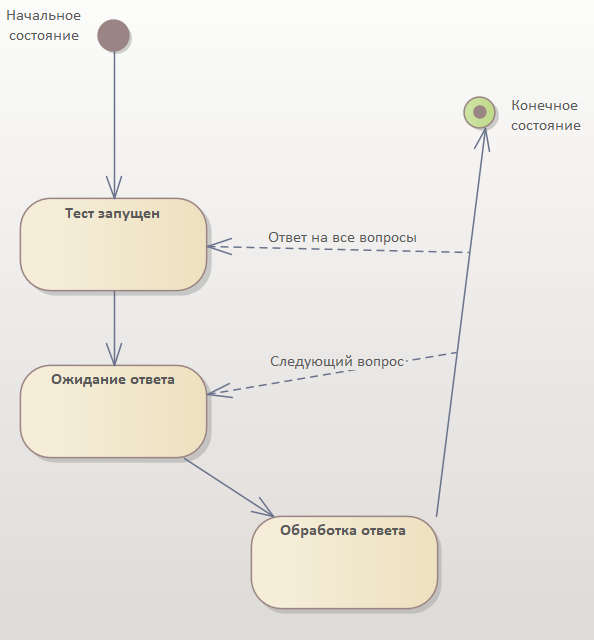


Рис. 2.5 – UML-диаграмма состояний

Описание диаграммы состояний:

1 Начальное состояние. Система переходит в это состояние после выбора пользователем теста и запуска процесса его прохождения.

2 Состояние «Тест запущен». В этом состоянии студенту отображается первый вопрос теста.

3 Состояние «Ожидание ответа». Система ждет, пока студент введет ответ на текущий вопрос. Это состояние завершается, как только ответ зарегистрирован.

4 Состояние «Обработка ответа». На этом этапе система проверяет правильность ответа и сохраняет результат. Если остались еще вопросы, система переходит обратно к состоянию «Тест запущен» и отображает следующий вопрос.

5 Конечное состояние. После завершения обработки всех вопросов система завершает тестирование, формирует итоговый результат и переходит в это состояние.

Диаграмма состояний наглядно демонстрирует жизненный цикл объекта «тестирование студента», начиная с запуска теста и заканчивая обработкой результатов. Такое представление позволяет анализировать возможные сценарии поведения системы и выявлять места, где могут быть допущены ошибки (например, неверная обработка ответа или сбои в переходах между состояниями).

Использование диаграммы состояний способствует разработке более стабильной системы, так как помогает понять, как объект ведет себя в зависимости от различных событий. В системе тестирования студентов это особенно важно для создания понятного и интуитивного интерфейса взаимодействия с пользователями.

# 2.3 Описание алгоритмов, реализующих бизнес-логику системы

Алгоритмы являются ключевым элементом бизнес-логики любой системы. Они определяют, как система обрабатывает данные, реагирует на пользовательские действия и обеспечивает выполнение поставленных задач.

В данной части представим основные алгоритмы автоматизированной системы тестирования студентов по языку C++. Были разработаны блок-схемы, описывающие логику работы программы в целом, а также основные методы: проверка ответов на вопросы теста и формирование итогового результата теста.

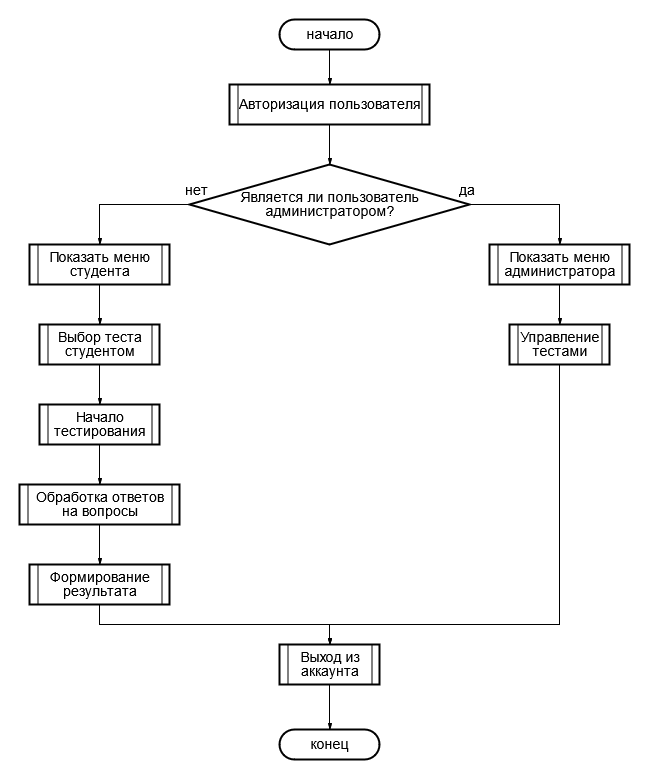


Рис. 2.6 – Схема алгоритма работы программы

Данный алгоритм демонстрирует, как система функционирует в зависимости от роли пользователя (администратор или студент). Включены этапы авторизации, выбора режима работы, тестирования и завершения программы.

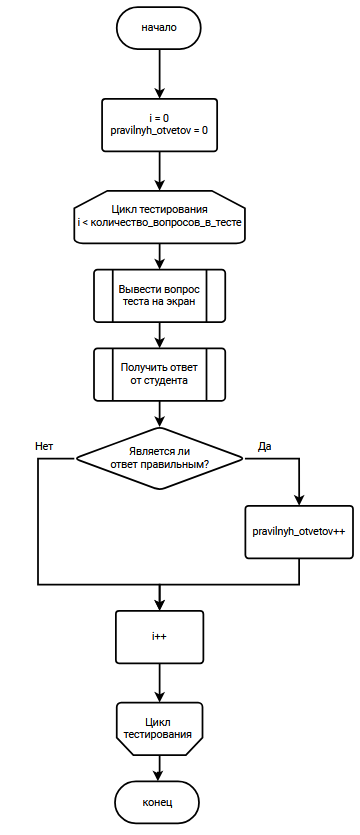


Рис. 2.7 – Схема алгоритма проверки ответа на вопрос

Алгоритм показывает логику обработки ответа студента. Система проверяет, совпадает ли ответ с правильным, обновляет статистику и переходит к следующему вопросу.

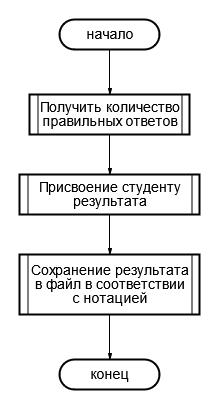


Рис. 2.8 – Схема алгоритма формирования результата

Этот алгоритм отвечает за анализ результатов теста. На основании правильных и неправильных ответов вычисляется итоговая оценка, которая сохраняется в системе.

Алгоритмы, реализующие бизнес-логику системы, обеспечивают ее функциональную целостность и корректное выполнение задач. Разработанные схемы наглядно иллюстрируют процессы работы системы в целом, а также ключевых методов, таких как проверка ответов и формирование результата. Их анализ позволяет выявить основные этапы обработки данных, что способствует оптимизации и совершенствованию системы. Разработанные алгоритмы легко расширяются, что делает систему масштабируемой и удобной для дальнейшего развития.

# 2.4 Описание созданных программных конструкций

Для реализации автоматизированной системы тестирования студентов по языку C++ использовались разнообразные возможности языка программирования C++, включая основные принципы объектно-ориентированного программирования, работу с потоками, обработку ошибок и использование стандартной библиотеки. Это позволило создать надежную, расширяемую и удобную в сопровождении программу. Рассмотрим примененные программные конструкции и их роль в реализации системы:

1 Реализация базовых принципов объектно-ориентированного программирования:

– инкапсуляция. В программе класс User демонстрирует инкапсуляцию. Поля login\_, password\_, is\_admin\_ сделаны закрытыми (private), а доступ к ним осуществляется через методы. Это защищает данные от прямого изменения извне и позволяет добавлять логику проверки:

std::string get\_login() const { return login\_; }

bool is\_administrator() const { return is\_admin\_; }

– наследование. В файле видно, что интерфейс IManageable наследуется классами Test и Result. Это позволяет реализовать общий интерфейс управления данными:

class Test : public IManageable {

public:

void load(std::istream& in) override;

void save(std::ostream& out) override;

};

– полиморфизм. Использование виртуальных функций в базовом классе IManageable позволяет по-разному обрабатывать объекты разных классов через указатель на базовый класс:

virtual void load(std::istream& in) = 0;

virtual void save(std::ostream& out) = 0;

2 Использование абстрактных классов. Класс IManageable является абстрактным, так как содержит чисто виртуальные функции. Он служит интерфейсом для классов Test и Result. Это позволяет унифицировать работу с разными типами данных (например, тесты и результаты):

class IManageable {

public:

virtual void load(std::istream& in) = 0;

virtual void save(std::ostream& out) = 0;

virtual ~IManageable() = default;

};

3 Передача параметров:

– по значению: Пример функции add\_question в классе Test. Здесь объект q передается по значению, а затем перемещается в вектор, что предотвращает лишнее копирование:

void add\_question(Question q) { questions\_.push\_back(std::move(q)); }

– по ссылке. Методы вроде authorize\_user используют передачу параметров по ссылке. Это экономит память и позволяет работать с оригинальным объектом:

bool authorize\_user(const std::shared\_ptr<User>& user);

4 Использование функций:

– пользовательские функции. Пример функции для поиска и сортировки тестов. Эта функция реализует специфичную логику, необходимую для работы с объектами тестов:

void search\_and\_sort\_tests();

– дружественные функции. В классе Question перегружены операторы >> и <<, объявленные дружественными. Это позволяет упрощать работу с потоками:

friend std::istream& operator>>(std::istream& is, Question& q);

friend std::ostream& operator<<(std::ostream& os, const Question& q);

– виртуальные функции. В классе IManageable используются виртуальные функции load и save, что позволяет полиморфно работать с разными типами объектов.

5 Пространства имен:

– встроенные. Например, пространство имен std активно используется для работы с потоками и контейнерами:

#include <iostream>

using namespace std;

– собственные. В программе существует собственное пространство имен app, что упрощает использование программы:

app::application app;

app.run();

6 Обработка ошибок. Блоки обработки исключений (try-catch) применяются для надежной работы с файлами и входными данными. Это гарантирует устойчивость программы при ошибках, таких как отсутствие файла:

try {

app.load\_results("results.txt");

} catch (const std::exception& ex) {

std::cerr << "Error: " << ex.what() << std::endl;

}

7 Перегрузка и переопределение:

– перегрузка операторов. Класс Question перегружает оператор << для удобного вывода:

std::ostream& operator<<(std::ostream& os, const Question& q) {

os << q.get\_text();

return os;

}

– переопределение методов. В классе Test переопределяется метод load базового класса IManageable:

void load(std::istream& in) override {

// Загрузка данных теста из потока

}

8 Шаблоны. В программе активно используются шаблоны, такие как std::vector и std::shared\_ptr. Они позволяют работать с универсальными контейнерами и указателями без написания дополнительного кода.

9 Динамическое выделение памяти и умные указатели. Используются умные указатели std::shared\_ptr для управления памятью. Это предотвращает утечки памяти:

std::vector<std::shared\_ptr<User>> users\_;

10 Потоки и перегрузка операторов ввода/вывода:

– работа с потоками. Используются потоки для чтения и записи данных:

std::ofstream out("results.txt");

app.save\_results(out);

– перегрузка операторов. Операторы ввода и вывода перегружены для работы с пользовательскими объектами (Question и Test).

В данной главе были рассмотрены основные программные конструкции, использованные в разработке проекта. Программа демонстрирует грамотное применение принципов ООП, обработку ошибок, работу с памятью через умные указатели, использование шаблонов и стандартной библиотеки C++. Это делает систему гибкой, надежной и легко расширяемой для новых требований и задач.

Для описания системы были разработаны и использованы UML-диаграммы различных типов, таких как диаграмма классов, диаграмма последовательностей и диаграмма состояний. Диаграмма классов позволила наглядно отразить архитектуру системы, включая взаимосвязи между компонентами и их ключевые атрибуты и методы. Диаграмма последовательностей показала взаимодействие объектов во времени, что позволило уточнить логику ключевых сценариев работы. Диаграмма состояний отразила переходы между состояниями объекта при изменении его контекста. Эти модели значительно упростили процесс проектирования системы и позволили избежать потенциальных ошибок на этапе разработки.

Помимо UML-моделирования, для описания информационной структуры системы была разработана модель IDEF1X, которая показала структуру данных, используемых в программе, и их взаимосвязь. Это помогло детально спроектировать механизмы работы с данными, включая чтение, запись и манипуляцию информацией, представленной в файлах.

В программной реализации системы активно использовались базовые принципы объектно-ориентированного программирования, такие как инкапсуляция, наследование и полиморфизм, что обеспечило гибкость и модульность кода. Использование абстрактных классов, шаблонов и умных указателей улучшило безопасность и удобство работы с памятью. Реализация дружественных функций, перегрузка операторов и методов, а также применение пространств имен способствовали упрощению структуры программы и улучшению ее читаемости.

Кроме того, в системе реализована обработка ошибок средствами языка C++, что делает ее устойчивой к некорректным данным или действиям пользователя [11]. Использование потоков ввода-вывода позволило организовать удобное взаимодействие с файлами, содержащими тесты, пользователей и результаты. Программа демонстрирует эффективное управление ресурсами, динамическое выделение памяти и применение современных подходов к программированию.

Таким образом, созданная система сочетает в себе эффективные методы проектирования, современные подходы к реализации и интуитивно понятный интерфейс. В результате, разработанная автоматизированная система тестирования является функциональной, надежной и расширяемой, что соответствует требованиям, поставленным в начале работы.

# 3 ОПИСАНИЕ АЛГОРИТМА ЗАПУСКА ПРИЛОЖЕНИЯ, ЕГО ИСПОЛЬЗОВАНИЯ, РЕЗУЛЬТАТЫ РАБОТЫ, ТЕСТИРОВАНИЯ ОБРАБОТКИ ОШИБОК

# 3.1 Алгоритм запуска приложения

Для того, чтобы воспользоваться приложением на другом компьютере, изначально необходимо скомпилировать проект в исполняемый файл платформы Windows (.exe). Это автоматически делает встроенный в IDE Visual Studio 2022 компилятор clang. Затем, в подпапке x64 проекта можно найти исполняемый файл. Его использование является безопасным для другого компьютера.

Затем, вместе с исполняемым файлом необходимо передать все текстовые файлы, в которых хранятся данные о пользователях, тестах и результатах. Очень важно, чтобы исполняемый файл и текстовые данные лежали в одной директории.

Для запуска программы на другом компьютере, на нем должны быть установлены VC++ Runtime библиотеки (программы на C++, созданные посредством Visual Studio, требуют наличия этих библиотек). Они не нужны на нашем компьютере, так как устанавливаются во время установки Visual Studio, однако не устанавливаются по умолчанию на компьютере конечного пользователя [10].

Затем, достаточно просто запустить исполняемый файл и начать пользоваться приложением.

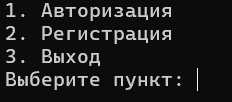


Рис. 3.1 – Скриншот главного окна приложения

# 3.2 Руководство пользователя

В данной части представлено руководство пользователя программы автоматизированной системы тестирования студентов по языку C++. Здесь представлены инструкции по осуществлению входа или регистрации в систему. Особое внимание уделяется процедуре аутентификации, необходимой для доступа к функционалу программы. Для наглядности и удобства пользователей представлен скриншот интерфейса (рисунки 3.2, 3.3), демонстрирующий меню авторизации, этапы ввода учетных данных и процесса входа в систему.

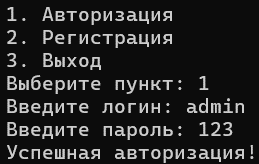


Рис. 3.2 – Скриншот процесса авторизации

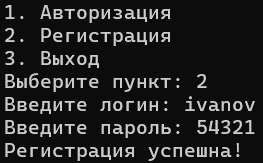


Рис. 3.3 – Скриншот процесса регистрации

Рассмотрим функционал администратора. При авторизации от имени администратора на экране будет отображено меню возможных действий, включающее в себя такие пункты как «Управление тестами», «Просмотр результатов», «Поиск и сортировка тестов», «Фильтровать тесты» и «Назад».

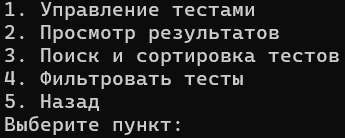


Рис. 3.4 – Скриншот меню администратора

Начнем с пункта «Управление тестами». В этом пункте предоставлены возможности по проведению CUD-работы с тестами. После завершения работы предоставляется возможность вернуться обратно к меню администратора.

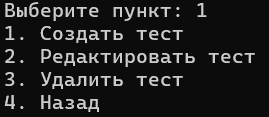


Рис. 3.5 – Скриншот пункта «Управление тестами»

Далее «Просмотр результатов». При выборе этого пункта меню администратор может просмотреть результаты всех тестов, которые когда-либо были пройдены всеми студентами. После завершения работы пользователя возвращает обратно в меню администратора.

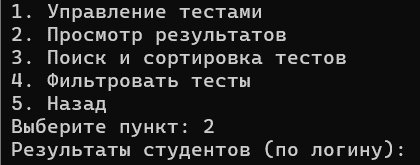


Рис. 3.6 – Скриншот пункта «Просмотр результатов»

Разберем пункт «Поиск и сортировка тестов». При выборе этого пункта меню администратор может отсортировать или произвести поиск тестов по заданному параметру. После завершения работы пользователя возвращает обратно в меню администратора.

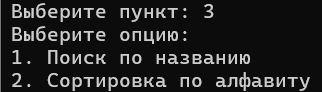


Рис. 3.7 – Скриншот пункта «Поиск и сортировка тестов»

Следующий пункт – «Фильтровать тесты». При выборе этого пункта меню администратор может отфильтровать тесты по заданному параметру. После завершения работы пользователя возвращает обратно в меню администратора.

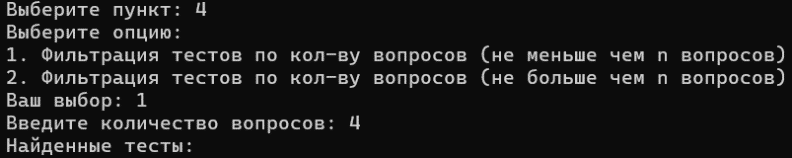


Рис. 3.8 – Скриншот пункта «Фильтровать тесты»

При выборе пункта «Назад» производится выход из аккаунта и возвращение к главному меню программы.

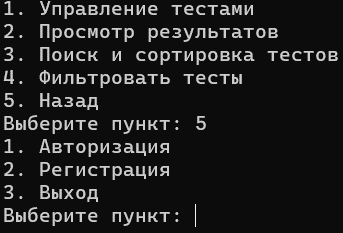


Рис. 3.9 – Скриншот пункта «Назад»

Рассмотрим функционал студента. При авторизации от имени студента на экране будет отображено меню возможных действий, включающее в себя такие пункты как «Пройти тест», «Просмотреть свои результаты», «Поиск и сортировка тестов», «Фильтровать тесты» и «Назад».

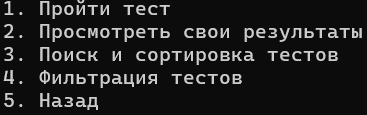


Рис. 3.10 – Скриншот меню студента

Начнем с «Пройти тест». При выборе этого пункта студент выбирает тест, который он хочет пройти, после чего он даёт ответ на все предлагаемые вопросы. Затем, после прохождения теста, ему будет показан его результат. После этого студента вернет обратно к меню студента.

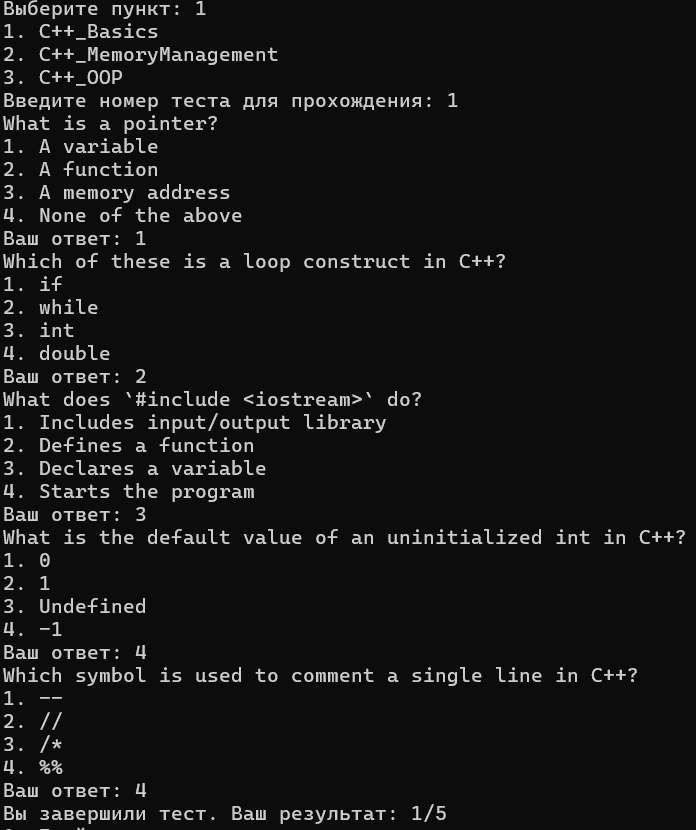


Рис. 3.11 – Скриншот пункта «Пройти тест»

Далее «Просмотреть свои результаты». При выборе этого пункта меню студент может просмотреть результаты всех тестов, которые когда-либо были им пройдены. После этого студента вернет обратно к меню студента.

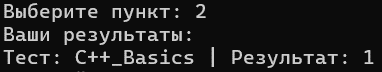


Рис. 3.12 – Скриншот пункта «Просмотреть свои результаты»

Пункты «Поиск и сортировка тестов», «Фильтрация тестов» и «Назад» полностью соответствуют тому же самому функционалу, что и в меню администратора.

После завершения работы с программой предоставляется возможность выйти из программы, выбрав пункт «Выход». Выход с помощью этого пункта предотвращает неправильное сохранение данных, их потерю и какие-либо дальнейшие ошибки при использовании разработанного ПО.

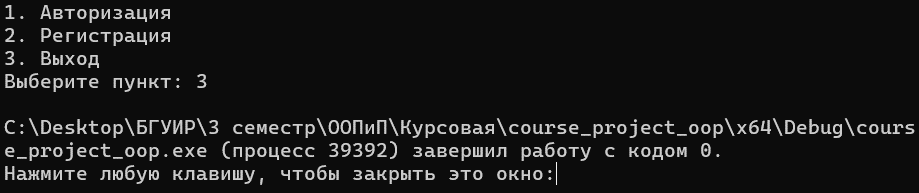


Рис. 3.13 – Скриншот пункта «Выход»

# 3.3 Тестирование работы приложения

В процессе тестирования работы приложения была проведена проверка обработки исключительных ситуаций [12]. Были смоделированы различные ошибки ввода данных, отсутствия или повреждения файлов, некорректных действий пользователя и проверена реакция программы [14].

Рассмотрим обработанные ошибки.

Ошибка: Некорректный ввод в меню. Если пользователь вводит нечисловое значение или некорректный пункт меню, программа выводит сообщение об ошибке и предлагает повторить ввод.

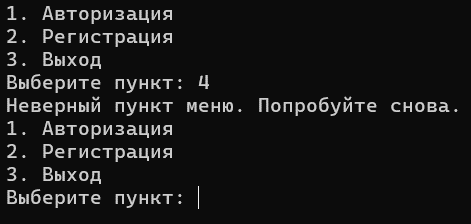


Рис. 3.14 – Обработка ошибки «Некорректный ввод в меню»

Ошибка: Отсутствие файла. При отсутствии файла с данными программа выводит сообщение об ошибке.



Рис. 3.15 – Обработка ошибки «Отсутствие файла»

Ошибка: Пустой файл. Если файл с данными пуст, программа предупреждает об этом.



Рис. 3.16 – Обработка ошибки «Пустой файл»

Ошибка: Дублирование пользователя. При регистрации программа проверяет, существует ли уже пользователь с указанным логином. Если да, выводится сообщение.

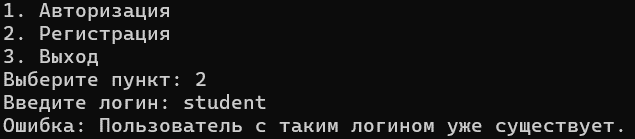


Рис. 3.17 – Обработка ошибки «Дублирование пользователя»

Ошибка: Неверный формат ввода вопроса или ответа. При создании тестов или вводе вопросов программа проверяет корректность данных. Если ввод некорректен, будет выведена ошибка.

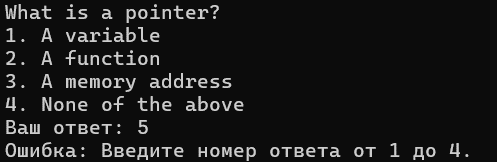


Рис. 3.18 – Обработка ошибки «Неверный формат ввода вопроса или ответа»

Ошибка: Выбор несуществующего теста. Если пользователь пытается выбрать тест, которого нет в списке, программа предупреждает.

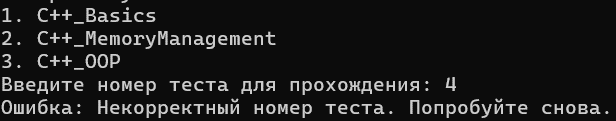


Рис. 3.19 – Обработка ошибки «Выбор несуществующего теста»

Ошибка: Неверные данные при загрузке файлов. При попытке загрузить данные из файла с некорректным форматом программа вызывает исключение.



Рис. 3.20 – Обработка ошибки «Неверные данные при загрузке файлов»

Тестирование показало, что приложение устойчиво к некорректным действиям пользователя, повреждённым файлам или отсутствию данных. Программа обрабатывает ошибки корректно, информирует пользователя о возникших проблемах и предлагает повторить действия [15]. Это обеспечивает удобство и безопасность использования.

Проведённое тестирование подтвердило, что приложение полностью соответствует требованиям:

– реализованы основные алгоритмы и сценарии использования;

– пользователи информируются об ошибках с помощью удобных и понятных сообщений;

– система устойчива к некорректному вводу, отсутствию данных и повреждённым файлам.

Эти результаты свидетельствуют о высокой надёжности и эффективности разработанной системы автоматизированного тестирования.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был проведён анализ и разработка программы для автоматизированного тестирования студентов по языку C++. Разработанная система позволила решить задачи, связанные с обработкой, хранением и представлением данных о тестах, пользователях и результатах. Программа полностью соответствует поставленным требованиям, обеспечивая поддержку множества пользователей, удобное управление тестами и сохранение результатов, а также устойчивость к ошибкам и некорректным действиям.

В первом разделе работы была изучена предметная область, связанная с процессом тестирования знаний студентов, включая особенности построения тестов и оценки их результатов. Были исследованы ключевые аспекты организации данных и построена модель IDEF0, которая позволила детально описать основные процессы системы. Модель отразила взаимодействие между различными элементами системы, что способствовало более точному пониманию и оптимизации её работы.

Во втором разделе была проведена разработка архитектуры системы, включая разработку модели IDEF1X, описывающую структуру файлов и взаимосвязь между элементами системы, и проектирование структуры системы с использованием UML-диаграмм. Диаграмма классов помогла определить основные элементы системы и их взаимосвязь, диаграмма последовательностей – проанализировать взаимодействие объектов, а диаграмма состояний – описать жизненный цикл ключевых процессов, таких как тестирование. Эти модели позволили задать чёткую архитектуру приложения, улучшив её надёжность и расширяемость.

В третьем разделе проводилось тестирование приложения, включая проверку обработки исключительных ситуаций. Были рассмотрены ошибки, связанные с некорректным вводом данных, отсутствием или повреждением файлов, а также дублированием записей. Программа показала высокую устойчивость и способность информировать пользователя о возникших проблемах, что повышает удобство и безопасность её использования.

В результате выполнения курсовой работы была достигнута поставленная цель – разработка автоматизированной системы тестирования студентов по языку C++. Созданная программа обеспечила удобство работы для преподавателей и студентов, сократив трудозатраты, связанные с созданием и проведением тестов, а также с анализом их результатов. Разработанное программное средство успешно интегрирует современные технологии C++ и проектирования ПО, что делает его готовым к дальнейшему использованию и расширению.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Глушаков, А.В. Основы автоматизации образовательных процессов / А.В. Глушаков. – Москва : Юрайт, 2018. – С. 12–18.

[2] Raj, A. Educational Automation: Principles and Practices / A. Raj. – New York : Springer, 2020. – P. 34–40.

[3] Anderson, L.W. A Taxonomy for Learning, Teaching, and Assessing / L.W. Anderson, D.R. Krathwohl. – New York : Longman, 2021. – P. 35–38.

[4] Moodle Documentation [Электронный ресурс]. – Режим доступа: https://docs.moodle.org. – Дата доступа: 30.09.2024.

[5] Blackboard Help Center [Электронный ресурс]. – Режим доступа: https://help.blackboard.com. – Дата доступа: 30.09.2024.

[6] Карпов, Д.В. Платформы для дистанционного обучения: сравнительный анализ / Д.В. Карпов, D.R. Krathwohl. – Москва : МГПУ, 2020. – С. 57–62.

[7] IDEF0. Знакомство с нотацией и пример использования [Электронный ресурс]. – Режим доступа: https://trinion.org/blog/idef0. – Дата доступа: 05.10.2024.

[8] Rumbaugh, J. Unified Modeling Language User Guide / J. Rumbaugh, D.R. Krathwohl. – New York : Addison-Wesley, 2005. – P. 145–155.

[9] Fowler, M UML Distilled: A Brief Guide to the Standard Object Modeling Language / M Fowler, D.R. Krathwohl. – New York : Addison-Wesley, 2004. – P. 50–55.

[10] Павловская, Т.А. C/C++. Программирование на языке высокого уровня: Учебник для вузов / Т.А. Павловская, D.R. Krathwohl. – Санкт-Петербург : Питер, 2012. – С. 321–330.

[11] Шилдт, Г. Язык программирования C++ для начинающих / Г. Шилдт, М. Вильямс. – Санкт-Петербург : Питер, 2021. – С. 89–95.

[12] Stroustrup, B. The C++ Programming Language / B. Stroustrup, М. Вильямс. – New York : Addison-Wesley, 2013. – P. 400–410.

[13] Google C++ Style Guide [Электронный ресурс]. – Режим доступа: https://google.github.io/styleguide/cppguide.html. – Дата доступа: 12.11.2024.

[14] Meyers, S. Effective C++: 55 Specific Ways to Improve Your Programs and Designs / S. Meyers, М. Вильямс. – New York : Addison-Wesley, 2005. – P. 210–220.

[15] Технология программирования [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/1195601. – Дата доступа: 29.11.2024.

# ПРИЛОЖЕНИЕ А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат»

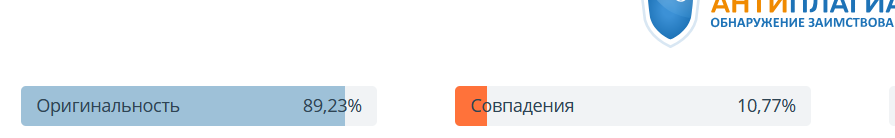


Рисунок А.1 – Проверка на Антиплагиат

# ПРИЛОЖЕНИЕ Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику

Функция кодирования пароля:

static std::string en\_decrypt\_password(const std::string& password)

{

const char key = static\_cast<char>(password.length() % 10);

std::string encrypted = password;

for (char& c : encrypted)

c ^= key;

return encrypted;

}

Функция прохождения теста:

void take\_test(const std::shared\_ptr<user>& user)

{

if (tests\_.empty())

{

std::cout << "Нет доступных тестов.\n";

return;

}

print\_tests();

std::cout << "Введите номер теста для прохождения: ";

size\_t index;

std::cin >> index;

if (std::cin.fail() || index < 1 || index > tests\_.size())

{

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

std::cout << "Ошибка: Некорректный номер теста. Попробуйте снова.\n";

return;

}

const auto& test = tests\_[index - 1];

int score = 0;

for (auto question : test->get\_questions())

{

std::cout << question.get\_text() << "\n";

const auto& answers = question.get\_answers();

for (size\_t i = 0; i < answers.size(); ++i)

std::cout << i + 1 << ". " << answers[i] << "\n";

std::cout << "Ваш ответ: ";

size\_t answer;

std::cin >> answer;

if (std::cin.fail() || answer < 1 || answer > answers.size())

{

std::cin.clear();

Продолжение приложения Б

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

std::cout << "Ошибка: Введите номер ответа от 1 до " << answers.size() << ".\n";

continue;

}

if (question.is\_correct(static\_cast<int>(answer) - 1))

++score;

}

std::cout << "Вы завершили тест. Ваш результат: " << score << "/" << test->get\_questions().size() << "\n";

results\_.push\_back(std::make\_shared<result>(user->get\_login(), test->get\_title(), score));

}

Функция считывания тестов из файла:

void load\_tests(const std::string& filename)

{

std::ifstream file(filename);

if (!file.is\_open())

{

std::cerr << "Ошибка: Не удалось открыть файл " << filename << ". Проверьте существование файла.\n";

return;

}

if (file.peek() == std::ifstream::traits\_type::eof())

{

std::cerr << "Ошибка: Файл " << filename << "пуст.\n";

return;

}

tests\_.clear();

try

{

while (file.peek() != EOF) {

auto test\_item = std::make\_shared<test>();

test\_item->load(file);

tests\_.push\_back(test\_item);

}

}

catch (const std::exception& ex)

{

std::cerr << "Ошибка при загрузке тестов: " << ex.what() << '\n';

}

}

Функция считывания пользователей из файла:

void load\_users(const std::string& filename)

{

std::ifstream file(filename);

if (!file.is\_open())

Продолжение приложения Б

{

std::cerr << "Ошибка: Не удалось открыть файл " << filename << ". Проверьте существование файла.\n";

return;

}

if (file.peek() == std::ifstream::traits\_type::eof())

{

std::cerr << "Ошибка: Файл " << filename << "пуст.\n";

return;

}

users\_.clear();

std::string login, password;

bool is\_admin;

while (file >> login >> password >> is\_admin)

users\_.push\_back(std::make\_shared<user>(login, password, is\_admin));

}

Функция регистрации нового студента в систему:

void register\_user()

{

std::string login, password;

std::cout << "Введите логин: ";

std::cin >> login;

if (login.empty())

{

std::cout << "Ошибка: Логин не может быть пустым. Попробуйте снова.\n";

return;

}

if (std::any\_of(users\_.begin(), users\_.end(), [&](const auto& user) { return user->get\_login() == login; }))

{

std::cout << "Ошибка: Пользователь с таким логином уже существует.\n";

return;

}

std::cout << "Введите пароль: ";

std::cin >> password;

for (const auto& user : users\_)

{

if (user->get\_login() == login)

{

std::cout << "Пользователь с таким логином уже существует.\n";

return;

}

}

users\_.push\_back(std::make\_shared<user>(login, user::en\_decrypt\_password(password), false));

std::cout << "Регистрация успешна!\n";

}

Продолжение приложения Б

Функция сохранения тестов в файл:

void save\_tests(const std::string& filename) const

{

std::ofstream file(filename);

if (!file.is\_open())

{

std::cerr << "Ошибка: Не удалось открыть файл для записи: " << filename << ".\n";

return;

}

try

{

for (const auto& test : tests\_)

test->save(file);

}

catch (const std::exception& ex)

{

std::cerr << "Ошибка при сохранении " << filename << ": " << ex.what() << '\n';

}

for (const auto& test : tests\_)

test->save(file);

}

Функция сохранения студентов в файл:

void save\_users(const std::string& filename) const

{

std::ofstream file(filename);

if (!file.is\_open())

{

std::cerr << "Ошибка: Не удалось открыть файл для записи: " << filename << ".\n";

return;

}

try

{

for (const auto& user : users\_)

file << user->get\_login() << ' '

<< user->get\_encrypted\_password() << ' '

<< (user->is\_administrator() ? 1 : 0) << '\n';

}

catch (const std::exception& ex)

{

std::cerr << "Ошибка при сохранении " << filename << ": " << ex.what() << '\n';

}

}

Функция считывания результатов из файла:

void load\_results(const std::string& filename)

{

std::ifstream file(filename);

if (!file.is\_open())

Продолжение приложения Б

{

std::cerr << "Ошибка: Не удалось открыть файл " << filename << ". Проверьте существование файла.\n";

return;

}

if (file.peek() == std::ifstream::traits\_type::eof())

{

std::cerr << "Ошибка: Файл " << filename << "пуст.\n";

return;

}

results\_.clear();

while (!file.eof())

{

auto res = std::make\_shared<result>("", "", 0);

try

{

res->load(file);

results\_.push\_back(res);

}

catch (const std::exception& ex)

{

std::cerr << "Ошибка при загрузке результата: " << ex.what() << '\n';

}

}

}

Функция сохранения результатов в файл:

void save\_results(const std::string& filename) const

{

std::ofstream file(filename);

if (!file.is\_open())

{

std::cerr << "Ошибка: Не удалось открыть файл для записи: " << filename << ".\n";

return;

}

try

{

for (const auto& result : results\_)

result->save(file);

}

catch (const std::exception& ex)

{

std::cerr << "Ошибка при сохранении " << filename << ": " << ex.what() << '\n';

}

}