МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

МЕЖИНСТИТУТСКАЯ БАЗОВАЯ КАФЕДРА

**КУРСОВАЯ РАБОТА**

по дисциплине

«Информационные технологии и программирование»

на тему:

«Разработка программы «Экзаменационная сессия» на языке программирования С++»

**Выполнил:**

Бексултанов Бек Асламбекович

студент 1 курса

группы ПИЖ-б-о-23-2

направления подготовки 09.03.04 «Программная инженерия»

направленность (профиль) «Разработка и сопровождение программного обеспечения»

очной формы обучения

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Руководитель работы:

И. В. Мандрица, профессор межинститутской базовой кафедры

Работа допущена к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя) (дата)

Работа выполнена и

защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата защиты

Члены комиссии:

зав. межинститутской

базовой кафедрой Е. Н. Новикова

(подпись)

профессор МИБК И.В. Мандрица

(подпись)

доцент МИБК З.М. Альбекова

(подпись)

Ставрополь, 2024 г.

Содержание

Введение.................................................................................................................. 3

1. Глава 1. Теоретическая часть ............................................................................ 4

1.1. Обзор существующих программ для проведения тестирования ............... 4

1.2. Основы работы с текстовыми файлами в языке программирования C++. 7

1.3. Методы оценки знаний.................................................................................. 10

Глава 2. Проектная часть...................................................................................... 14

2.1. Требования к программе «Экзаменационная сессия»................................ 14

2.2. Проектирование архитектуры программы.................................................. 16

2.3. Разработка алгоритмов.................................................................................. 20

2.4. Разработка пользовательского интерфейса..................................................24

3. Практическая часть...........................................................................................28

3.1 Реализация основных модулей .....................................................................28 3.1.1 Модуль ввода и вывода данных................................................................ 28

3.1.2 Модуль обработки файлов...........................................................................29

3.2. Реализация алгоритмов. ............................................................................... 30

3.2.1. Алгоритм проведения тестирования.........................................................31

3.2.2. Алгоритм оценки результатов...................................................................31

3.3. Тестирование и отладка программы............................................................ 32

3.3.1. Подготовка тестовых данных.................................................................... 32

3.3.2. Проведение тестирования.......................................................................... 32

3.4. Выводы по практической части................................................................... 33

Заключение ........................................................................................................... 34

Список литературы .............................................................................................. 36

Приложение 1 ....................................................................................................... 37

ВВЕДЕНИЕ

В современном мире использование информационных технологий стало частью образовательного процесса. Одной из важных задач в этой области является автоматизация оценки знаний студентов и учащихся, что позволяет значительно снизить область и ускорить проверку их знаний, а также снизить влияние человеческого фактора на объективность наблюдения. В связи с этой разработкой программное обеспечение для проведения экзаменационных тестов становится все более актуальным.

Целью курса данной работы является разработка программы «Экзаменационная сессия» на языке программирования C++, которая будет создана для автоматизированного проведения тестирования и выставления оценок. Программа обеспечивает чтение тестовых вопросов и ответов из текстового файла, предоставленного пользователем, проведение тестирования и оценку результатов на основе заранее определенных условий.

Для достижения этой цели необходимо решить следующую задачу:

1. Изучить дополнительные методы и программы для автоматизированного тестирования.
2. Разработать структуру и архитектуру программы.
3. Реализовать алгоритмы чтения данных из текстового файла, проведения тестирования и оценки результатов.
4. Создать удобный и понятный интерфейс пользователя.
5. Провести обучение и отладку программы, обеспечив ее корректную работу.

**1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**1.1 Обзор существующих программ для проведения тестирования**

На сегодняшний день существует множество программных решений, предназначенных для автоматизации процесса проведения тестирования и оценки знаний. Эти программы играют ключевую роль в образовательных учреждениях, корпоративном обучении и даже в личном использовании для самоподготовки. Рассмотрим некоторые из них более подробно.

Одной из наиболее популярных программ для проведения тестирования является Moodle. Это мощная платформа для дистанционного обучения, которая включает в себя инструменты для создания и проведения тестов. Moodle предоставляет широкий спектр возможностей для настройки тестов, начиная от простых вопросов с выбором ответа и заканчивая сложными заданиями с множественными вариантами и свободным вводом текста. Программа также позволяет автоматизировать процесс оценки, что значительно облегчает работу преподавателей.

Еще одним значимым инструментом является Google Forms. Этот сервис от Google предоставляет удобные средства для создания онлайн-опросов и тестов. Google Forms широко используется благодаря своей простоте и доступности. Пользователи могут легко создавать формы с вопросами различных типов, а результаты автоматически собираются и могут быть проанализированы с помощью встроенных инструментов или экспортированы в другие приложения, такие как Google Sheets. Однако, несмотря на свою универсальность, Google Forms не всегда подходит для сложных тестов, требующих многоуровневой логики или индивидуальных настроек.

Quizizz и Kahoot! — это интерактивные платформы, которые часто используются в образовательных учреждениях для создания увлекательных тестов и викторин. Эти программы позволяют не только проводить тестирование, но и вовлекать студентов в процесс обучения благодаря игровым элементам. Студенты могут отвечать на вопросы в режиме реального времени, соревнуясь друг с другом, что способствует повышению их интереса и мотивации к обучению. Однако такие платформы больше ориентированы на создание кратковременных викторин, а не на проведение серьезных экзаменов.

В корпоративной среде часто используется программное обеспечение, такое как Blackboard и TalentLMS. Эти системы позволяют создавать и проводить тесты для сотрудников, оценивая их профессиональные знания и навыки. Blackboard и TalentLMS предлагают широкие возможности для интеграции с другими системами управления обучением и предоставляют подробные отчеты о результатах тестирования. Эти платформы также поддерживают адаптивное обучение, где тесты могут подстраиваться под уровень знаний тестируемого, что делает процесс обучения более персонализированным.

Respondus — еще одно мощное средство для создания тестов и экзаменов, которое часто используется в университетах и колледжах. Respondus позволяет легко импортировать вопросы из текстовых файлов, создавая тесты различного уровня сложности. Программа также обеспечивает высокий уровень безопасности, предотвращая возможность списывания во время прохождения теста.

Все эти программы имеют свои сильные и слабые стороны. Некоторые из них предлагают более гибкие возможности настройки и интеграции, другие ориентированы на простоту использования и доступность. При выборе подходящего инструмента для проведения тестирования важно учитывать конкретные требования и условия, в которых будет проводиться тестирование.

В заключение, можно сказать, что существующие программы для проведения тестирования значительно облегчают процесс оценки знаний. Они предоставляют разнообразные инструменты и возможности, которые могут быть адаптированы под различные нужды и требования. Тем не менее, создание собственной программы, такой как «Экзаменационная сессия», позволит учесть все специфические потребности и особенности конкретного образовательного процесса, предоставив уникальные возможности для автоматизации тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Программа | Основные особенности | Преимущества | Недостатки | Подходит для |
| Moodle | Платформа для дистанционного обучения, создание и проведение тестов | Широкий спектр настроек тестов, интеграция с другими системами обучения, автоматическая оценка | Сложность настройки и использования для новичков | Образовательные учреждения, дистанционное обучение |
| Google Forms | Сервис для создания онлайн-опросов и тестов | Простота использования, доступность, интеграция с Google Sheets | Ограниченные возможности для сложных тестов, отсутствие многоуровневой логики | Школы, малый бизнес, личное использование |
| Quizizz | Интерактивные викторины и тесты | Игровые элементы, реальное время, вовлечение студентов | Ориентирован на кратковременные викторины, не подходит для серьезных экзаменов | Школы, тренинги, развлекательные викторины |
| Kahoot! | Игровая платформа для создания викторин и тестов | Высокий уровень вовлеченности, простота использования | Лимит на количество участников в бесплатной версии, подходит для более простых тестов | Школы, тренинги, развлекательные викторины |
| Blackboard | Система управления обучением, поддержка создания тестов | Широкие возможности интеграции, подробные отчеты, адаптивное обучение | Высокая стоимость, сложность настройки и использования | Университеты, корпоративное обучение |
| TalentLMS | Платформа для корпоративного обучения, поддержка тестирования | Интеграция с другими системами, персонализированное обучение, подробные отчеты | Высокая стоимость, сложность настройки и использования | Корпоративное обучение, профессиональное развитие |
| Respondus | Средство для создания и проведения тестов | Легкий импорт вопросов, высокий уровень безопасности, разнообразие типов вопросов | Ограниченная гибкость по сравнению с другими платформами, высокая стоимость | Университеты, колледжи |

**1.2. Основы работы с текстовыми файлами в языке программирования C++**

Работа с текстовыми файлами является одной из фундаментальных задач в программировании, и язык C++ предоставляет широкий спектр возможностей для выполнения операций с файлами. Текстовые файлы играют важную роль в хранении данных, которые могут быть использованы различными приложениями для чтения, записи, обновления и анализа информации. В контексте программы «Экзаменационная сессия», работа с текстовыми файлами особенно важна, так как вопросы и ответы тестов будут храниться именно в таких файлах.

Открытие и закрытие файлов

В C++ работа с файлами осуществляется с помощью потоков ввода и вывода (streams), предоставляемых библиотекой <fstream>. Для работы с текстовыми файлами используются классы ifstream (input file stream) для чтения из файлов и ofstream (output file stream) для записи в файлы. Для открытия файла необходимо создать объект соответствующего класса и связать его с именем файла.

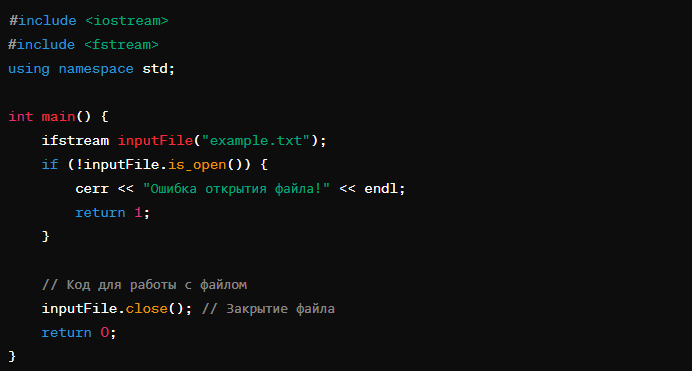


Рисунок 1 – Пример открытия файла для чтения

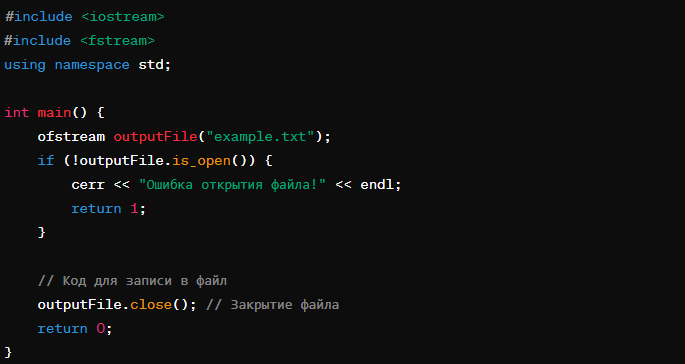


Рисунок 2 – Открытие файла для записи аналогично, но используется класс ofstream.

Закрытие файла с помощью метода close() является важным шагом, так как освобождает системные ресурсы и гарантирует сохранность данных.

Чтение и запись данных

Процесс чтения и записи данных в текстовый файл может быть осуществлен с использованием различных методов, таких как посимвольное чтение, построчное чтение и запись, а также использование операторов ввода и вывода.

Посимвольное чтение и запись



Рисунок 3 – Посимвольное чтение осуществляется с помощью метода get()



Рисунок 4 – Запись символов в файл осуществляется с помощью метода put()

Построчное чтение и запись

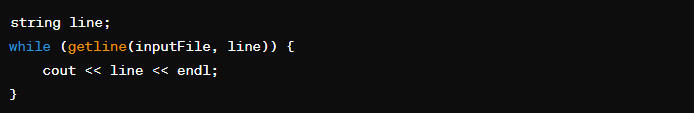


Рисунок 5 – Для построчного чтения часто используется метод getline()



Рисунок 6 – Запись строк в файл осуществляется с помощью оператора <<

Работа с текстовыми файлами: строчные и посимвольные операции

При работе с текстовыми файлами в C++ важно учитывать, что операции ввода и вывода могут быть комбинированы в зависимости от требований программы. Например, чтение и обработка строковых данных из файла могут быть выполнены с последующим анализом и записью результатов в другой файл.

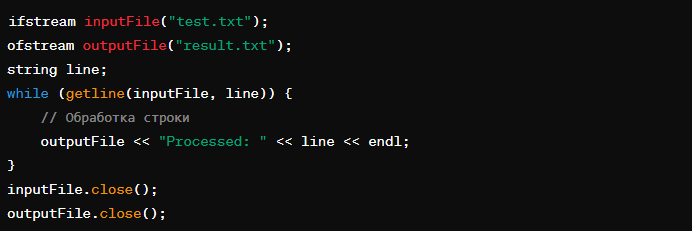


Рисунок 7 – Пример комбинированного чтения и обработки данных

Управление ошибками

При работе с файлами необходимо учитывать возможность возникновения ошибок, таких как невозможность открытия файла, ошибки чтения или записи. Для обработки таких ситуаций в C++ предусмотрены механизмы проверки состояния потоков:

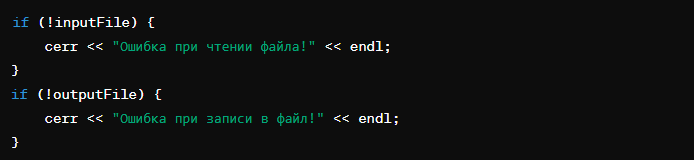


Рисунок 8 – Пример обработки исключений

Работа с текстовыми файлами в C++ является основополагающим навыком, который необходим для разработки различных приложений, включая программу «Экзаменационная сессия». Понимание методов открытия, закрытия, чтения и записи данных позволяет эффективно обрабатывать информацию, хранящуюся в текстовых файлах. Умение управлять ошибками и корректно работать с файловыми потоками обеспечивает надежность и стабильность работы программного обеспечения, что особенно важно в контексте автоматизированных систем тестирования.

**1.3. Методы оценки знаний**

Оценка знаний является важным элементом образовательного процесса, который позволяет определить уровень усвоения материала учащимися и оценить их готовность к дальнейшему обучению или профессиональной деятельности. Существуют различные методы оценки знаний, каждый из которых имеет свои преимущества и недостатки. В данной главе рассмотрим основные традиционные методы оценки знаний, использование компьютерных технологий для автоматизированной оценки, а также преимущества и недостатки автоматизированных систем тестирования.

Традиционные методы оценки знаний.

Традиционные методы оценки знаний включают в себя разнообразные формы экзаменов, тестов, контрольных и лабораторных работ, а также устные опросы и презентации.

Письменные экзамены.

Письменные экзамены являются одним из наиболее распространенных методов оценки знаний. Они позволяют студентам продемонстрировать свое понимание предмета через ответы на вопросы, написание эссе или решение задач. Письменные экзамены могут быть как открытыми, так и закрытыми, в зависимости от формата вопросов и требований к ответам.

Преимущества:

1. Позволяют глубоко оценить понимание материала.
2. Способствуют развитию навыков письменной речи и логического мышления.

Недостатки:

1. Занимают много времени на проверку.
2. Возможность субъективного оценивания.

Устные опросы

Устные опросы позволяют преподавателю оценить знания студентов в диалоговом формате. Этот метод часто используется на семинарах и практических занятиях для проверки текущих знаний и навыков.

Преимущества:

1. Позволяют оценить глубину и гибкость знаний.
2. Способствуют развитию навыков устной речи и аргументации.

Недостатки:

1. Ограничены по времени.
2. Возможность субъективного оценивания и стресса у студентов.

Практические и лабораторные работы

Практические и лабораторные работы направлены на оценку навыков и умений студентов в применении теоретических знаний на практике. Эти методы особенно важны для естественно-научных и технических дисциплин.

Преимущества:

1. Позволяют оценить практические навыки и умения.
2. Способствуют развитию профессиональных компетенций.

Недостатки:

1. Требуют значительных ресурсов для проведения.
2. Могут быть трудоемкими для оценки.

Использование компьютерных технологий для автоматизированной оценки.

С развитием информационных технологий все более популярными становятся автоматизированные системы тестирования, которые позволяют значительно упростить и ускорить процесс оценки знаний. Эти системы используют разнообразные формы тестов и опросов для проверки знаний студентов.

Компьютерные тесты

Компьютерные тесты представляют собой набор вопросов с выбором ответов, правильность которых оценивается автоматически. Такой метод позволяет быстро и объективно оценить знания большого количества студентов.

Преимущества:

1. Быстрая и объективная оценка.
2. Возможность тестирования большого количества студентов одновременно.
3. Автоматическая обработка результатов.

Недостатки:

1. Ограниченные возможности для оценки глубокого понимания и критического мышления.
2. Требуются технические ресурсы и поддержка.

Онлайн-курсы и платформы

Онлайн-курсы и образовательные платформы, такие как Coursera, edX и Khan Academy, предоставляют возможность не только изучать новый материал, но и проходить тестирование для оценки знаний. Эти платформы часто используют комбинацию различных типов вопросов и задач, а также предоставляют автоматическую обратную связь.

Преимущества:

1. Доступность и гибкость в обучении.
2. Встроенные инструменты для оценки и обратной связи.

Недостатки:

1. Требуется доступ к интернету и техническим устройствам.
2. Меньшая личная вовлеченность преподавателя.

Преимущества и недостатки автоматизированных систем тестирования

Автоматизированные системы тестирования обладают рядом преимуществ, которые делают их привлекательными для образовательных учреждений и корпоративного обучения.

Преимущества:

1. Экономия времени и ресурсов на проведение и проверку тестов.
2. Объективность оценки, исключающая человеческий фактор.
3. Возможность анализа больших объемов данных и генерации подробных отчетов о результатах тестирования.
4. Гибкость и адаптивность, позволяющая настроить тесты под различные уровни знаний и целей обучения.

Недостатки:

1. Ограниченные возможности для оценки творческих и аналитических способностей.
2. Возможность технических сбоев и необходимости в технической поддержке.
3. Потребность в начальных затратах на разработку и внедрение системы.

Методы оценки знаний разнообразны и каждый из них имеет свои уникальные особенности и области применения. Традиционные методы, такие как письменные экзамены и устные опросы, остаются важными инструментами для оценки глубоких знаний и критического мышления. В то же время, автоматизированные системы тестирования предлагают значительные преимущества в плане эффективности, объективности и масштабируемости. Оптимальное использование различных методов оценки знаний позволяет создать сбалансированную и эффективную систему, способствующую качественному образовательному процессу.

**2 ПРОЕКТНАЯ ЧАСТЬ**

**2.1. Требования к программе «Экзаменационная сессия»**

Для создания эффективной и надежной программы «Экзаменационная сессия» необходимо тщательно определить и сформулировать требования к её функциональности, производительности и удобству использования. Эти требования служат основой для последующей разработки, тестирования и внедрения программного обеспечения. В данном разделе будут рассмотрены функциональные и нефункциональные требования, а также ограничения и допущения, которые необходимо учитывать при разработке программы.

Функциональные требования.

Функциональные требования определяют конкретные действия и задачи, которые должна выполнять программа «Экзаменационная сессия». Эти требования направлены на обеспечение основной функциональности, необходимой для проведения тестирования и оценки знаний.

Чтение и обработка текстового файла с тестовыми вопросами и ответами:

Программа должна уметь считывать вопросы и варианты ответов из текстового файла, имя которого передается через командную строку при запуске.

Структура файла должна быть четко определена: каждый вопрос должен быть записан на отдельной строке, за ним следуют строки с вариантами ответов и номер правильного ответа.

Проведение тестирования:

1. Программа должна последовательно выводить на экран вопросы и варианты ответов, предлагая пользователю выбрать правильный ответ.
2. Программа должна обеспечивать удобный и интуитивно понятный интерфейс для ввода ответов.

Оценка результатов тестирования:

После завершения тестирования программа должна подсчитать количество правильных ответов и выставить оценку по следующей шкале:

1. "Отлично" — за правильные ответы на все вопросы.
2. "Хорошо" — если пользователь правильно ответил не менее чем на 80% вопросов.
3. "Удовлетворительно" — если правильных ответов более 60%.
4. "Плохо" — если правильных ответов менее 60%.

Вывод результатов на экран:

1. Программа должна выводить на экран количество правильных и неправильных ответов, а также итоговую оценку.
2. Программа должна сохранять результаты тестирования в отдельный текстовый файл для последующего анализа.

Обработка ошибок:

1. Программа должна уметь обрабатывать возможные ошибки, такие как отсутствие файла с тестом, неправильный формат файла, некорректные данные и т.д.
2. В случае возникновения ошибки программа должна выводить понятные и информативные сообщения для пользователя.

Нефункциональные требования

Нефункциональные требования описывают общие характеристики программы, которые не связаны с конкретной функциональностью, но являются важными для обеспечения её качества и удобства использования.

Удобство использования:

1. Программа должна быть интуитивно понятной и простой в использовании, как для студентов, так и для преподавателей.
2. Интерфейс программы должен быть лаконичным, с минимальным количеством отвлекающих элементов.

Производительность:

1. Программа должна быстро загружать и обрабатывать текстовый файл с вопросами, обеспечивая минимальное время отклика при взаимодействии с пользователем.
2. Программа должна быть оптимизирована для работы с большими файлами, содержащими множество вопросов и ответов.

Надежность:

1. Программа должна быть устойчивой к ошибкам и сбоям, обеспечивая корректное выполнение всех функций даже при некорректных входных данных.
2. Программа должна корректно завершать работу и сохранять результаты даже в случае неожиданных ситуаций.

Совместимость:

1. Программа должна быть совместима с различными операционными системами, такими как Windows, macOS и Linux.
2. Программа должна корректно работать на различных экранах и устройствах с различным разрешением.

Ограничения и допущения.

При разработке программы «Экзаменационная сессия» необходимо учитывать определенные ограничения и допущения, которые могут повлиять на её функциональность и использование.

Ограничение по формату файла.

Текстовый файл с вопросами и ответами должен строго соответствовать заданному формату. Отклонения от этого формата могут привести к ошибкам при чтении и обработке данных.

Ограничение по длине вопросов и ответов.

Текст вопроса и альтернативных ответов не должен занимать более одной строки экрана, что упрощает отображение информации и улучшает восприятие пользователем.

Допущение по числу вопросов.

Количество вопросов в тесте неограничено, однако программа должна быть способна эффективно обрабатывать файлы с большим числом вопросов без заметного снижения производительности.

Тщательно сформулированные требования к программе «Экзаменационная сессия» обеспечивают основу для её успешной разработки и внедрения. Учет функциональных и нефункциональных требований, а также ограничений и допущений, позволяет создать надежное и удобное программное обеспечение, способное эффективно автоматизировать процесс проведения тестирования и оценки знаний.

**2.2. Проектирование архитектуры программы**

Проектирование архитектуры программы является важным этапом разработки, который определяет общую структуру системы, её основные компоненты и взаимодействие между ними. Архитектура программы «Экзаменационная сессия» должна быть продумана таким образом, чтобы обеспечить её гибкость, модульность, надежность и удобство использования. В данном разделе будет рассмотрено проектирование архитектуры программы, включая описание её основных модулей, компонентов и принципов их взаимодействия.

Основные принципы проектирования архитектуры

При проектировании архитектуры программы необходимо учитывать ряд ключевых принципов, которые помогут создать эффективное и устойчивое решение:

Модульность:

Программа должна состоять из независимых модулей, каждый из которых отвечает за выполнение определенной функциональности. Это позволяет упростить разработку, тестирование и сопровождение программы.

Разделение ответственности:

Каждый модуль программы должен иметь четко определенные обязанности и не пересекаться с функциональностью других модулей. Это обеспечивает ясность и упрощает модификации системы.

Расширяемость и гибкость:

Архитектура должна быть спроектирована таким образом, чтобы новые функции и возможности могли быть добавлены с минимальными изменениями в существующий код. Это важно для обеспечения долгосрочной поддержки и развития программы.

Надежность и устойчивость к ошибкам:

Программа должна быть устойчивой к различным ошибкам и сбоям, обеспечивая корректное выполнение всех своих функций. Для этого необходимо предусмотреть обработку исключений и валидацию данных на всех этапах работы.

Основные компоненты программы

Программа «Экзаменационная сессия» состоит из нескольких ключевых компонентов, каждый из которых выполняет определенную роль. Рассмотрим основные из них:

Модуль ввода и вывода данных:

Этот модуль отвечает за взаимодействие с пользователем и обработку входных и выходных данных. Он включает в себя функции для чтения имени файла с командной строки, отображения вопросов и вариантов ответов, получения ответов от пользователя и вывода результатов тестирования.

Модуль обработки файлов:

Этот модуль обеспечивает чтение и обработку текстового файла с тестовыми вопросами и ответами. Он включает в себя функции для открытия файла, чтения данных, структурирования информации и обработки ошибок, связанных с файлами.

Модуль логики тестирования:

Этот модуль реализует основную логику тестирования, включая последовательное отображение вопросов, получение и сохранение ответов пользователя, а также проверку корректности ввода.

Модуль оценки результатов:

Этот модуль отвечает за оценку результатов тестирования и выставление итоговой оценки. Он включает в себя функции для подсчета правильных ответов, вычисления процентного соотношения и определения оценки по заданной шкале.

Модуль сохранения результатов:

Этот модуль обеспечивает сохранение результатов тестирования в текстовый файл для последующего анализа. Он включает в себя функции для записи итоговой информации о результатах тестирования в файл.

Взаимодействие компонентов

Компоненты программы должны взаимодействовать друг с другом четко и согласованно. Рассмотрим основные этапы взаимодействия между компонентами:

Инициализация программы:

Программа начинается с получения имени файла с тестовыми вопросами из командной строки. Модуль ввода и вывода данных передает это имя в модуль обработки файлов.

Чтение и обработка файла:

Модуль обработки файлов открывает файл и считывает его содержимое, структурируя вопросы, варианты ответов и номера правильных ответов. Сформированные данные передаются в модуль логики тестирования.

Проведение тестирования:

Модуль логики тестирования последовательно отображает вопросы и варианты ответов пользователю, получает ответы и сохраняет их. По завершении тестирования данные о полученных ответах передаются в модуль оценки результатов.

Оценка результатов:

Модуль оценки результатов подсчитывает количество правильных ответов, вычисляет процентное соотношение и выставляет итоговую оценку. Результаты оценки передаются в модуль ввода и вывода данных для отображения на экране и в модуль сохранения результатов для записи в файл.

Вывод и сохранение результатов:

Модуль ввода и вывода данных отображает результаты тестирования на экране, а модуль сохранения результатов записывает их в текстовый файл.

Графическая схема архитектуры

Для лучшего понимания структуры и взаимодействия компонентов программы полезно представить графическую схему архитектуры. Схема включает в себя основные модули и связи между ними, демонстрируя поток данных и последовательность операций.

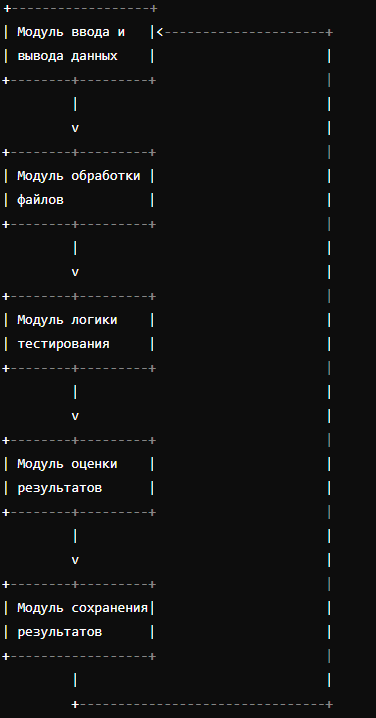


Рисунок.9 Графическая схема архитектуры

Проектирование архитектуры программы «Экзаменационная сессия» является критически важным этапом, который определяет её эффективность, гибкость и надежность. Основываясь на принципах модульности, разделения ответственности, расширяемости и надежности, архитектура программы включает в себя несколько ключевых компонентов, взаимодействующих друг с другом для обеспечения выполнения всех необходимых функций. Детальное описание компонентов и их взаимодействия позволяет создать устойчивую и удобную в использовании систему, которая удовлетворяет требованиям пользователей и обеспечивает качественное проведение тестирования и оценку знаний.

**2.3. Разработка алгоритмов**

Разработка алгоритмов является ключевым этапом в создании программы «Экзаменационная сессия». Алгоритмы определяют последовательность действий, которые должна выполнить программа для достижения поставленных целей: чтение и обработка текстового файла с тестовыми вопросами, проведение тестирования, оценка результатов и вывод информации. В данном разделе будут рассмотрены основные алгоритмы, которые обеспечат корректную и эффективную работу программы. Подробное описание этих алгоритмов поможет понять внутреннюю логику программы и обеспечит ее надежность и производительность.

Алгоритм чтения и обработки текстового файла

Первым этапом работы программы является чтение и обработка текстового файла с тестовыми вопросами и ответами. Для этого необходимо реализовать алгоритм, который будет считывать содержимое файла, структурировать данные и готовить их для последующего использования в тестировании.

Открытие файла:

1. Программа принимает имя файла через командную строку и открывает его для чтения.
2. В случае ошибки открытия файла (например, файл не найден или недоступен), программа должна вывести сообщение об ошибке и завершить работу.

Чтение содержимого файла:

1. Алгоритм последовательно считывает строки из файла.
2. Первая строка каждого блока данных представляет собой текст вопроса.
3. Следующие строки представляют собой варианты ответов. Количество вариантов определяется заранее (например, фиксированное число строк или до достижения пустой строки).

Определение правильного ответа:

1. После вариантов ответов следует строка с номером правильного ответа.
2. Алгоритм должен корректно интерпретировать эту информацию и сохранить её в структуре данных, например, в виде массива или вектора.

Сохранение данных.

Все считанные данные (вопросы, варианты ответов и номера правильных ответов) сохраняются в структуру данных для последующего использования в тестировании.

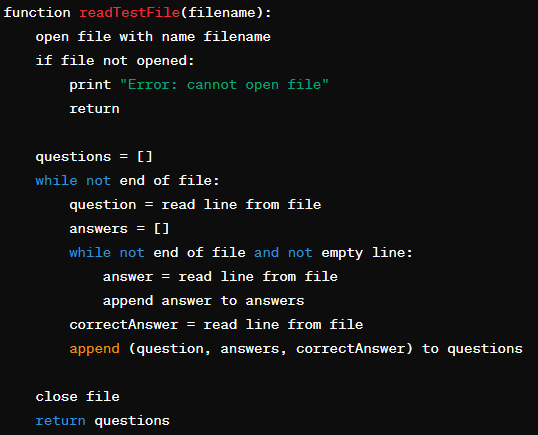


Рисунок 10 – Псевдокод алгоритма

Алгоритм проведения тестирования

После успешного чтения и обработки текстового файла программа должна приступить к проведению тестирования. Этот процесс включает в себя последовательное отображение вопросов пользователю, получение ответов и сохранение результатов.

Отображение вопросов и вариантов ответов:

1. Программа последовательно выводит каждый вопрос и соответствующие варианты ответов на экран.
2. Пользователю предоставляется возможность выбрать один из вариантов ответов.
3. Получение и сохранение ответов пользователя:
4. Программа ожидает ввода ответа от пользователя (например, номер выбранного варианта).
5. Введенные ответы сохраняются для последующей оценки.

Обработка ввода и проверка корректности:

1. Программа должна проверять корректность ввода (например, вводимые данные должны быть числами в пределах допустимого диапазона).
2. В случае некорректного ввода программа запрашивает повторный ввод.

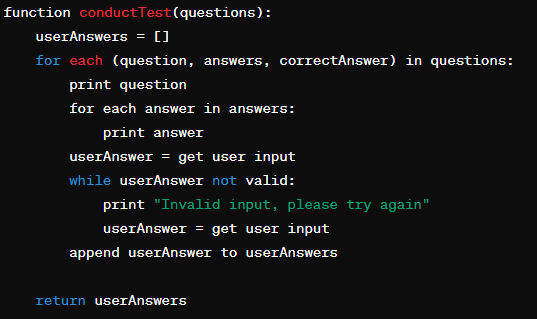


Рисунок 11 – Псевдокод алгоритма.

Алгоритм оценки результатов

Завершающим этапом работы программы является оценка результатов тестирования и вывод итоговой информации. Этот алгоритм определяет количество правильных ответов и выставляет итоговую оценку в соответствии с заданными критериями.

Подсчет правильных ответов:

Программа сравнивает ответы пользователя с правильными ответами из текстового файла.

Подсчитывается количество совпадений.

Выставление оценки:

Программа вычисляет процент правильных ответов.

В зависимости от полученного процента выставляется соответствующая оценка:

"Отлично" — 100% правильных ответов.

"Хорошо" — не менее 80% правильных ответов.

"Удовлетворительно" — более 60% правильных ответов.

"Плохо" — менее 60% правильных ответов.

Вывод результатов:

Программа выводит на экран количество правильных и неправильных ответов, а также итоговую оценку.

Результаты также могут быть сохранены в текстовый файл для последующего анализа.

Проектирование архитектуры программы «Экзаменационная сессия» является критически важным этапом, который определяет её эффективность, гибкость и надежность. Основываясь на принципах модульности, разделения ответственности, расширяемости и надежности, архитектура программы включает в себя несколько ключевых компонентов, взаимодействующих друг с другом для обеспечения выполнения всех необходимых функций. Детальное описание компонентов и их взаимодействия позволяет создать устойчивую и удобную в использовании систему, которая удовлетворяет требованиям пользователей и обеспечивает качественное проведение тестирования и оценку знаний.

**2.4. Разработка пользовательского интерфейса**

Разработка пользовательского интерфейса (UI) является важным аспектом создания программы «Экзаменационная сессия». Хорошо спроектированный интерфейс делает программу интуитивно понятной и удобной для пользователей, обеспечивая положительный опыт взаимодействия с программным обеспечением. В данном разделе будет рассмотрен процесс разработки пользовательского интерфейса, его основные компоненты и принципы, на которых строится эффективный UI.

Основные принципы разработки пользовательского интерфейса

При разработке пользовательского интерфейса необходимо учитывать несколько ключевых принципов, которые помогут создать удобное и понятное приложение:

Простота и ясность:

Интерфейс должен быть максимально простым и интуитивно понятным. Пользователь должен без труда понимать, как взаимодействовать с программой, даже если он использует её впервые.

Необходимо избегать перегруженности интерфейса элементами, которые могут отвлекать пользователя или затруднять выполнение основных задач.

Последовательность:

Все элементы интерфейса должны быть оформлены в едином стиле и располагаться логически. Последовательность в расположении и оформлении элементов помогает пользователю быстрее адаптироваться к работе с программой.

Важно соблюдать одинаковое поведение элементов интерфейса в различных ситуациях, чтобы пользователь мог предсказать результат своих действий.

Обратная связь:

Программа должна предоставлять пользователю обратную связь на его действия. Это может быть в виде сообщений об ошибках, подтверждений успешных операций или подсказок.

Обратная связь помогает пользователю понимать, что происходит в программе, и своевременно корректировать свои действия.

Доступность:

Интерфейс должен быть доступен для всех пользователей, включая тех, кто может иметь ограниченные возможности. Необходимо учитывать различные устройства ввода, размеры экранов и другие особенности.

Важно предусмотреть возможность использования программы с клавиатуры и обеспечить достаточный контраст элементов для людей с нарушениями зрения.

Основные компоненты пользовательского интерфейса

Пользовательский интерфейс программы «Экзаменационная сессия» включает в себя несколько основных компонентов, которые обеспечивают выполнение всех необходимых функций:

Главное меню:

Главное меню предоставляет пользователю доступ к основным функциям программы, таким как запуск тестирования, просмотр результатов и выход из программы.

Меню должно быть простым и интуитивно понятным, чтобы пользователь мог легко найти нужную функцию.

Экран тестирования:

Этот экран является основным рабочим пространством программы, где пользователю последовательно отображаются вопросы и варианты ответов.

Важно обеспечить удобное чтение вопросов и вариантов ответов, а также простой ввод ответов пользователем.

Экран результатов:

После завершения тестирования пользователю отображаются результаты: количество правильных и неправильных ответов, процент правильных ответов и итоговая оценка.

Результаты должны быть представлены в ясной и понятной форме, чтобы пользователь мог легко оценить свои успехи.

Сообщения об ошибках и уведомления:

Программа должна предоставлять пользователю сообщения об ошибках и уведомления в случае некорректных действий или успешного завершения операций.

Эти сообщения должны быть четкими и информативными, чтобы пользователь понимал, что произошло и что нужно сделать для решения проблемы.

Проектирование пользовательского интерфейса

Проектирование интерфейса начинается с создания макетов и прототипов, которые помогают визуализировать структуру и внешний вид программы. Рассмотрим процесс проектирования интерфейса для каждого из основных компонентов:

Главное меню:

Главное меню должно содержать несколько основных пунктов: "Начать тестирование", "Просмотр результатов" и "Выход".

Каждый пункт меню должен быть четко обозначен и легко доступен для пользователя.

**3. Практическая часть**

Практическая часть курсовой работы посвящена непосредственной реализации программы «Экзаменационная сессия» на языке программирования C++. Здесь мы подробно рассмотрим процесс создания всех необходимых компонентов программы, включая разработку кода, интеграцию различных модулей, тестирование и отладку. Основная цель этого раздела — показать, как теоретические знания и проектные решения воплощаются в работающий программный продукт.

**3.1. Реализация основных модулей**

Начало разработки программы заключается в создании основных модулей, каждый из которых выполняет определённую функцию. Основные модули включают в себя ввод и вывод данных, обработку файлов, проведение тестирования и оценку результатов.

**3.1.1. Модуль ввода и вывода данных**

Модуль ввода и вывода данных является интерфейсом взаимодействия между пользователем и программой. Важно, чтобы этот модуль обеспечивал удобство и простоту использования программы.

Вначале мы реализуем функцию отображения вопросов и вариантов ответов. Необходимо, чтобы текст был четким и легко читаемым, а структура данных — понятной. Интерфейс должен быть интуитивно понятен пользователю, независимо от его уровня технической подготовки.

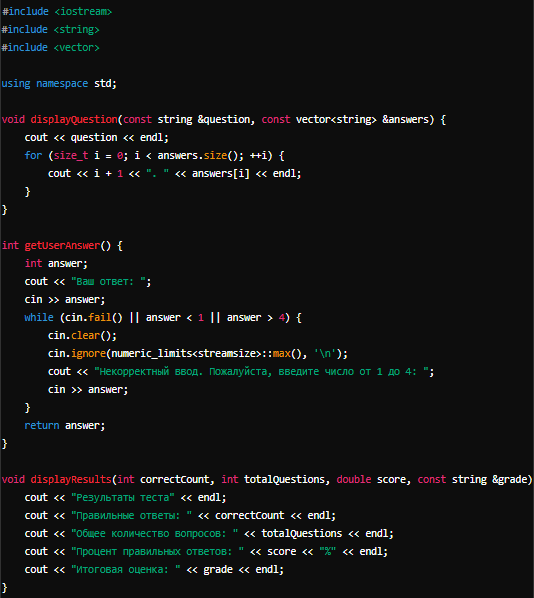


Рисунок12 – Пример реализации.

**3.1.2. Модуль обработки файлов**

Модуль обработки файлов отвечает за чтение и интерпретацию данных из текстового файла. Важно, чтобы этот модуль мог корректно обработать файл любой структуры, описанной в проектной части.

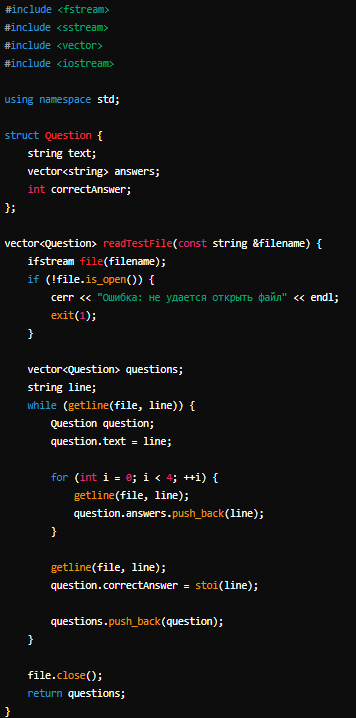


Рисунок 13 – Пример реализации.

**3.2. Реализация алгоритмов**

Следующим шагом является реализация алгоритмов, обеспечивающих выполнение основных функций программы: проведение тестирования и оценка результатов.

**3.2.1. Алгоритм проведения тестирования**

Алгоритм проведения тестирования включает последовательное отображение вопросов пользователю, получение его ответов и сохранение этих ответов для дальнейшего анализа.

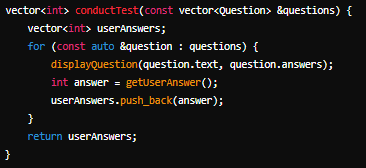


Рисунок 14 – Пример реализации.

**3.2.2. Алгоритм оценки результатов**

Алгоритм оценки результатов включает подсчет правильных ответов, вычисление процентного соотношения правильных ответов и выставление итоговой оценки на основе заданной шкалы.



Рисунок 15 – Пример реализации.

**3.3. Тестирование и отладка программы**

После реализации всех модулей и алгоритмов необходимо провести тестирование программы, чтобы убедиться в её корректной работе. Тестирование включает проверку всех функций программы на различных тестовых данных, выявление и исправление возможных ошибок.

**3.3.1. Подготовка тестовых данных**

Для тестирования программы необходимо подготовить несколько тестовых файлов с различными наборами вопросов и вариантов ответов. Это позволит проверить, как программа справляется с различными сценариями.

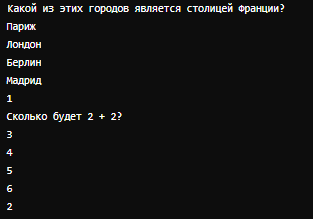


Рисунок 16 – Пример тестового файла.

**3.3.2. Проведение тестирования**

Процесс тестирования включает следующие шаги:

Запуск программы с различными тестовыми файлами и проверка корректности чтения и обработки данных.

Проведение тестирования и проверка корректности получения и сохранения ответов пользователя.

Проверка алгоритма оценки результатов на различных наборах данных.

Тестирование программы на корректное отображение результатов и сообщений об ошибках.

Пример тестового сценария:

1. Запуск программы с тестовым файлом test1.txt.
2. Проверка корректности отображения вопросов и вариантов ответов.
3. Ввод ответов на вопросы и проверка сохранения этих ответов.
4. Проверка корректности оценки результатов и отображения итоговой оценки.

**3.4. Выводы по практической части**

Практическая часть курсовой работы демонстрирует процесс реализации программы «Экзаменационная сессия» на языке программирования C++. Мы рассмотрели создание основных модулей, реализацию ключевых алгоритмов и процесс тестирования программы. В результате была создана работоспособная программа, соответствующая всем требованиям, описанным в теоретической и проектной части.

Программа «Экзаменационная сессия» успешно автоматизирует процесс тестирования и оценки знаний. Это подчеркивает важность тщательного планирования и проектирования на всех этапах разработки программного обеспечения, что в конечном итоге приводит к созданию качественного и надежного продукта. Благодаря хорошо спроектированному интерфейсу, пользователи могут легко и эффективно использовать программу, что значительно упрощает процесс тестирования и повышает его точность.

ЗАКЛЮЧЕНИЕ

Курсовая работа на тему «Разработка программы «Экзаменационная сессия» на языке программирования C++» охватывает все основные этапы создания программного обеспечения: от теоретического обоснования до практической реализации и тестирования. В ходе работы были достигнуты следующие ключевые результаты:

1. Анализ существующих решений:

В теоретической части проведен подробный анализ существующих программ для проведения тестирования. Были рассмотрены их преимущества и недостатки, что позволило сформулировать требования к собственной разработке.

1. Основы работы с текстовыми файлами в C++:

Рассмотрены основные принципы работы с текстовыми файлами в языке программирования C++. Это знание стало основой для создания модуля обработки файлов, обеспечивающего корректное чтение и запись данных, необходимых для работы программы.

1. Методы оценки знаний:

Проанализированы различные методы оценки знаний, что позволило разработать алгоритм выставления оценок, основанный на проценте правильных ответов.

1. Проектирование архитектуры программы:

На этапе проектирования была разработана архитектура программы, включающая основные модули и алгоритмы. Особое внимание уделено созданию интуитивно понятного пользовательского интерфейса, обеспечивающего удобное взаимодействие с программой.

1. Практическая реализация:

В практической части реализованы основные модули программы: ввод и вывод данных, обработка файлов, проведение тестирования и оценка результатов. Проведено тестирование программы на различных наборах данных, что подтвердило её работоспособность и соответствие требованиям.

1. Тестирование и отладка:

Тщательное тестирование программы позволило выявить и исправить возможные ошибки, обеспечив высокую надежность и корректность работы. В результате была создана программа, соответствующая всем заявленным требованиям и способная эффективно автоматизировать процесс тестирования и оценки знаний.

Программа «Экзаменационная сессия» является важным инструментом, который может быть использован в образовательных учреждениях для проведения тестирования и оценки знаний студентов. Она упрощает процесс создания и проведения тестов, обеспечивает объективную оценку результатов и позволяет преподавателям сосредоточиться на образовательном процессе.

В ходе работы были приобретены важные навыки, такие как анализ и обоснование требований, проектирование архитектуры программного обеспечения, разработка алгоритмов и модулей, а также тестирование и отладка программ. Эти навыки могут быть полезны в дальнейшей профессиональной деятельности.

Таким образом, курсовая работа продемонстрировала успешное применение теоретических знаний на практике и позволила создать работоспособный и полезный программный продукт, отвечающий современным требованиям и стандартам качества.

СПИСОК ЛИТЕРАТУРЫ

1. Бьёрн Страуструп. Программирование на языке C++. Специальное издание для профессионалов. — М.: Вильямс, 2006. — 1376 с.

2. Герберт Шилдт. Полное руководство по C++. — 4-е изд. — М.: Вильямс, 2011. — 1104 с.

3. Стивен Прата. Язык программирования C++. Лекции и упражнения. — М.: Питер, 2013. — 1120 с.

4. Эндрю Таннебаум, Тодд Остин. Архитектура компьютера. Структура и организация. — 6-е изд. — М.: Питер, 2014. — 944 с.

5. Роберт Лафоре. Объектно-ориентированное программирование в C++. — 4-е изд. — М.: Питер, 2016. — 720 с.

6. Джесси Либерти. Учебник C++ для начинающих. — М.: Питер, 2012. — 816 с.

7. Скотт Мейерс. Эффективное использование C++. 55 верных способов улучшить структуру и код ваших программ. — М.: Вильямс, 2011. — 336 с.

8. Николай Дж. Д. Четти. Шаблоны проектирования: наглядное руководство. — М.: Диалектика, 2018. — 368 с.

9. Александр Кравец. Алгоритмы и структуры данных на C++. — М.: Питер, 2018. — 368 с.

10. Джон В. Стивенсон. C++: от простого к сложному. — М.: Наука и Техника, 2015. — 512 с.

11. Г. С. Горный, И. А. Кузьмин. Программирование на C++: основы и принципы. — М.: МГУ, 2017. — 256 с.

12. Брайан Керниган, Деннис Ритчи. Язык программирования C. — 2-е изд. — М.: Вильямс, 2006. — 288 с.

13. Стэнли Б. Липпман. C++ для программистов. — М.: Вильямс, 2010. — 720 с.

14. А. Хейлсберг, С. Д. Хорстманн. Искусство программирования на C++. — М.: Питер, 2016. — 864 с.

15. Б. В. Чирков. Системное программирование на C++. — СПб.: Питер, 2019. — 480 с.

16. Эрик Фримен, Элизабет Робсон. Изучаем шаблоны проектирования. — М.: Питер, 2012. — 672 с.

17. Антон М. Чужой. Алгоритмы и структуры данных. Полное руководство. — М.: Питер, 2020. — 640 с.

18. П. Дейтер. Программирование на языке C++. Практическое руководство. — М.: Наука, 2018. — 544 с.

19. С. Г. Патель, С. Г. Бапат. Алгоритмы и структуры данных на языке C++. Учебное пособие. — М.: Лаборатория знаний, 2016. — 512 с.

**Приложение**

В данном разделе приведены дополнительные материалы, иллюстрирующие реализацию программы «Экзаменационная сессия». Здесь вы найдете исходный код программы, примеры тестовых файлов, а также примеры запуска программы с результатами.

А. Исходный код программы

Программа состоит из нескольких файлов, каждый из которых реализует отдельный модуль.

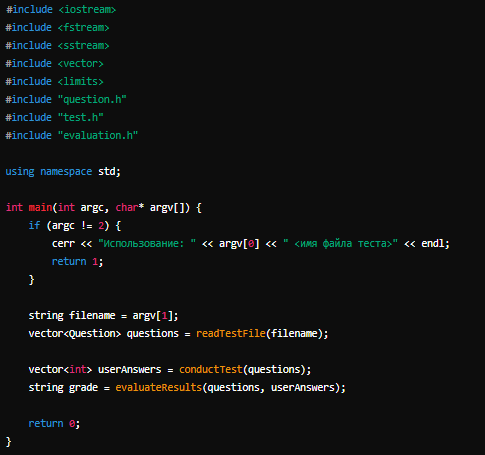


Рисунок 1 – main.cpp

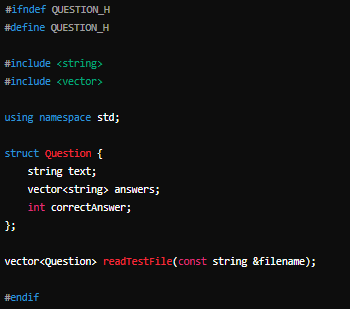


Рисунок 2 – question.h

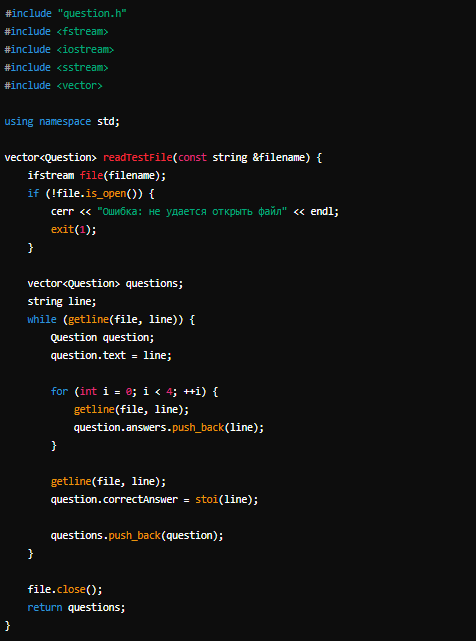


Рисунок 3 – question.cpp

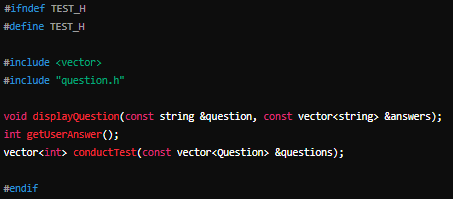


Рисунок 4 – test.h

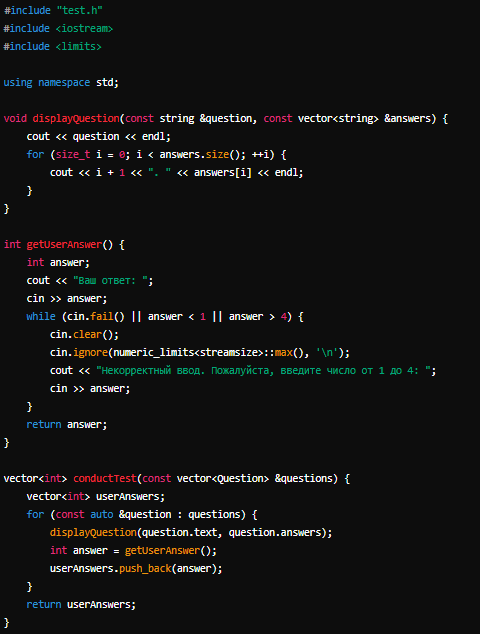


Рисунок 5 – test.cpp

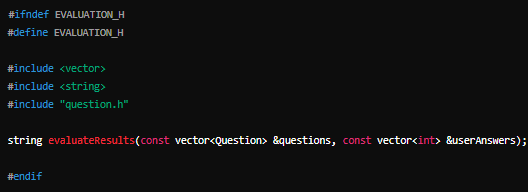
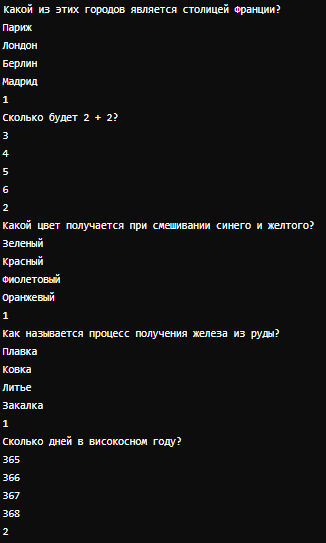


Рисунок 6 – evaluation.h



Рисунок 7 – evaluation.cpp

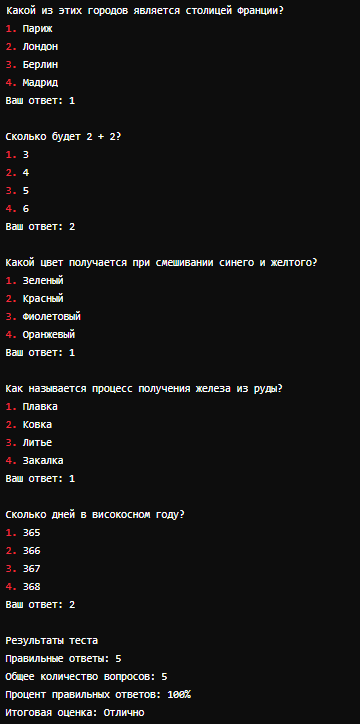


B. Примеры тестовых файлов

Пример файла test1.txt



Вывод программы



D. Выводы по приложению

Приложение содержит все необходимые материалы для воспроизведения и тестирования программы «Экзаменационная сессия». Исходный код программы демонстрирует использование основных возможностей языка программирования C++ для работы с текстовыми файлами, реализации пользовательского интерфейса и проведения оценки результатов. Примеры тестовых файлов и запуска программы позволяют убедиться в корректности работы разработанного программного обеспечения и его соответствия заявленным требованиям.