Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Технологии разработки программного обеспечения»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Ассистент кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Карпейчик |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«Разработка программы учета УСПЕВАЕМОСТИ СТУДЕНТА»**

БГУИР КР 1-40 05 01-02 001 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 272303  БАДЕЙ Павел Юрьевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**РЕФЕРАТ**

БГУИР КР 1-40 05 01-02 001 ПЗ

**Бадей, П.Ю.** Разработка программы учета успеваемости студента/              П.Ю. Бадей. – Минск: БГУИР, 2023. – 46 с.

Пояснительная записка 46 с., 28 рис., 8 источников, 4 приложения

УЧЕТ УСПЕВАЕМОСТИ СТУДЕНТОВ, ОТОБРАЖЕНИЕ СРЕДНЕЙ ОЦЕНКИ НА КОНТРОЛЬНОЙ ТОЧКЕ, СОРТИРОВКА ОЦЕНОК ЗА ЭКЗАМЕН, МОДЕЛИ *UML*, *IDEF*0, *BPMN*, СХЕМЫ АЛГОРИТМОВ, ПРОГРАММНОЕ СРЕДСТВО.

*Цель* *курсовой работы*: систематизация и углубление теоретических знаний, полученных за время обучения, а так же приобретение и закрепление навыков самостоятельной работы. Программа «Учет успеваемости студентов» должна служить для оперативного просмотра данных о студентах. В данной курсовой работе будет реализовано приложение, которое позволит осуществлять самоконтроль студента просматривая собственный электронный журнал успеваемости.

*Методология проведения работы*: в процессе решения поставленных задач использованы принципы системного подхода и аналитические методы.

*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 нотаций IDEF0 и BPMN; 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*: c помощью разработанного программного средства можно проанализировать успеваемость студента учебного заведения. Такое приложение призвано дисциплинировать студента в процессе обучения, способствует контролю собственной успеваемости.

Разработанное программное средство полностью отвечает всем функциональным требованиям, необходимым при планировании, учете и анализе складских операций на предприятии.

**СОДЕРЖАНИЕ**

[1 Анализ и иоднлирование предметной области 6](#_Toc135604799)

[1.1 Описание предметной области 6](#_Toc135604800)

[1.2 Разработка функциональной модели предметной области в нотации IDEF0 7](#_Toc135604801)

[1.3 Разработка BPMN-модели основного процесса предметной области 10](#_Toc135604802)

[1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований. 11](#_Toc135604803)

[1.5 UML-модели представления программного средства и их описание 12](#_Toc135604804)

[2 Проектирование и конструирование программного средства 17](#_Toc135604805)

[2.1 Постановка задачи 17](#_Toc135604806)

[2.2 Разработка модульной структуры 17](#_Toc135604807)

[2.3 Выбор способа организации данных 19](#_Toc135604808)

[2.4 Разработка перечня пользовательских функций программы 21](#_Toc135604809)

[2.5 Разработка схем алгоритмов работы программы 22](#_Toc135604810)

[3 Тестирование и проверка работоспособности программного средства 27](#_Toc135604811)

[4 Инструкция по развертыванию приложения и сквозной пример 29](#_Toc135604812)

[4.1 Авторизация 29](#_Toc135604813)

[4.2 Модуль администратора 31](#_Toc135604814)

[4.3 Модуль пользователя 32](#_Toc135604815)

[Заключение 35](#_Toc135604816)

[Список использованных источников 36](#_Toc135604817)

[Приложение А (обязательное) Проверка на заимствование 37](#_Toc135604818)

[Приложение Б (обязательное) Листинг кода 38](#_Toc135604819)

[Приложение В (справочное) Схема общего алгоритма работы программы 44](#_Toc135604820)

[Приложение Г (обязательное) Ведомость документов курсовой работы 46](#_Toc135604821)

## 

**ВВЕДЕНИЕ**

Исторически сложилось так, что развитие общества не стоит на месте. Каждое столетие учеными совершаются какие-либо открытия и новые создаются новые изобретения. Недаром говорят, что наш 21 век – век информационных технологий. Мы живем в эпоху построения информационной цивилизации, которая рождается в результате развития и постоянного совершенствования информационно-компьютерных технологий.

Компьютеризация охватывает всё больше и больше жизненных процессов человека. Компьютерная техника развивается с каждым годом, устройства становятся компактнее, производительнее, надёжнее. На первое место становится удобство и простота использования. В наше время не нужно никаких специализированных знаний, чтобы пользоваться компьютером в повседневной жизни. Производители программных продуктов предлагают нам интуитивно-понятные интерфейсы.

Сегодня информационные технологии задействованы везде: в промышленности, в авиатранспорте, ж/д транспорте, науке, образовании, социальных структурах, государственном управлении, экономики и культуре.

Развитие компьютерных технологий позволило упростить ручную, монотонную и расчетную работу. На данный момент создано большое количество программ, обрабатывающих разнотипную информацию. В настоящее время работа любого предприятия просто немыслима без применения компьютерных технологий. Современные программы могут значительно упростить процесс управления предприятием и производством. И только с использованием компьютеров и соответствующих программ современное предприятие может плодотворно работать и достигать немыслимых при некомпьютерной обработке информации результатов.

Одной из важнейших задач университета является развитие у студентов умений самостоятельно управлять учебной деятельностью, контролировать и оценивать ее результаты, что, в свою очередь, позволит студенту эффективно управлять процессом овладения знаниями в процессе обучения.

В последние годы все более значимой становится проблема оптимизации обучения, возрастает роль управления учебно-воспитательным процессом. Проблема качества обучения и самоконтроль студента приобретает международное значение.

Во всех сферах деятельности человека информационные технологии развиваются быстрыми темпами. Осуществление самоконтроля успеваемости программными средствами является удобной и важной новизной в учебном процессе.

## 1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

### **1.1 Описание предметной области**

Область применения информационных технологий растет по мере развития общества и технического прогресса как в Беларуси, так и во всем мире в целом. Новые технологии в области информационных технологий непрерывно внедряются во все сферы деятельности человека, начиная от быта и учебы, заканчивая разработкой новых областей применения данных технологий.

Информационные технологии активно применяются в учебном процессе, как способ более эффективного обучения будущих специалистов, не только в области IT, но и для подготовки будущих пилотов, инженеров, экономистов, врачей, дизайнеров и других специалистов. К примеру, в медицине, реализация информационных технологий хорошо применяется в применении виртуальной реальности при работе хирурга и его обучении.

В современном мире без информационных технологий не может обходиться ни одно общество, предприятие или учреждение. В школах они применяются при построении баз данных учета успеваемости школьников, в средних профессиональных и высших учебных заведениях, более активно применяются информационные технологии при ведении учета успеваемости студентов, в банках данных хранятся научные работы.

В крупных городах и мегаполисах невозможно представить ни один день без использования информационных технологий, начиная с утренних новостей, идущих на современном телевизоре, и заканчивая цифровым будильником.

Любой человек, отказавшийся от использования не только информационных, но и большинства современных технологий, изобретенными человечеством за последние десятилетия, обречен на гораздо более трудное существование.

Технологии – это топливо для человечества, обеспечивающее наше движение к высшим целям. Конечно, не каждое топливо дает достаточную отдачу, иногда даже отрицательную. Но факт остается фактом: информационные технологии упрощают нашу повседневную жизнь, делают ее проще.

Развитие информационных технологий облегчает работу специалистов в различных областях. Таким образом, необходимость информационных технологий не подвергается сомнению и активно финансируется всеми предприятиями, государственными и частными структурами.

Больше половины современного рынка занимают информационные технологии, как в Беларуси, так и за рубежом. Большинство, уже больше половины операций расчета, таких как покупка ценных бумаг, продажа или покупка акций, общение, дистанционное обучение в учебных заведениях и онлайн-тренинги, просмотр и многие другие действия происходят через интернет, который относится к информационным технологиям.

Не стоит упускать из виду тот факт, что информационные технологии активно внедряются и в высших учебных заведениях. Данные информационные системы становятся незаменимым инструментом в обеспечении успешного и стабильного развития университета, снижая нагрузку с персонала учебного заведения, забирая на себя большую часть работы с документами и их хранения.

Предметная область "разработка программы учета успеваемости студента" относится к сфере образования и включает в себя процесс отслеживания и оценки учебных достижений студентов в учебных заведениях. Для эффективной организации образовательного процесса необходимо вести учет успеваемости каждого студента.

В образовательных учреждениях разработка учета успеваемости студента является важным аспектом управления учебным процессом, поскольку она позволяет более эффективно использовать ресурсы и повышать качество образования.

Целью деятельности образовательной организации является обеспечение современного качественного образования. Контроль успеваемости - комплексный процесс, в котором принимают участие учебная часть вуза, кафедры и деканат.

Одной из важнейших задач университета является развитие у студентов умений самостоятельно управлять учебной деятельностью, контролировать и оценивать ее результаты, что, в свою очередь, позволит студенту эффективно управлять процессом овладения знаниями в процессе обучения.

Учет успеваемости студента сформирует у студента самостоятельность к принятию решений о своих успехах в изучаемых предметах; позволит ответственно подходить к подготовке к тому или иному предмету, основываясь на ранее полученных оценках и баллах; даст возможность преподавателям действовать мобильнее за счет функций программы, которые предоставляют преподавателям фильтровать информацию о студентах по многим критериям.

### **1.2 Разработка функциональной модели предметной области в нотации IDEF0**

IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов [4]. Описание системы с помощью IDEF0 называется функциональной моделью. Функциональная модель предназначена для описания существующих бизнес-процессов, в котором используются как естественный, так и графический языки. Методология IDEF0 предписывает построение иерархической системы диаграмм - единичных описаний фрагментов системы.

На рисунке 1.1 представлена диаграмма IDEF0. На изображении описан основной блок “Отобразить успеваемость студента”. Входящие стрелки – “данные студента”, “аккаунт в мессенджере”, “архив отдела кадров”, “журнал группы”. Это то, что необходимо знать для начала работы.

Управляющие для автоматизации системы медицинского центра – “встроенный мессенджер”, “нормативные акты организации”, “инструкция по делопроизводству”, “протокол о порядке организации и проведении текущей аттестации студентов”.

В роли “Механизмов” администратор и преподаватель. После завершения процесса на выходе мы получаем “рейтинг студента” и “выставленные оценки/баллы/пересдачи”.

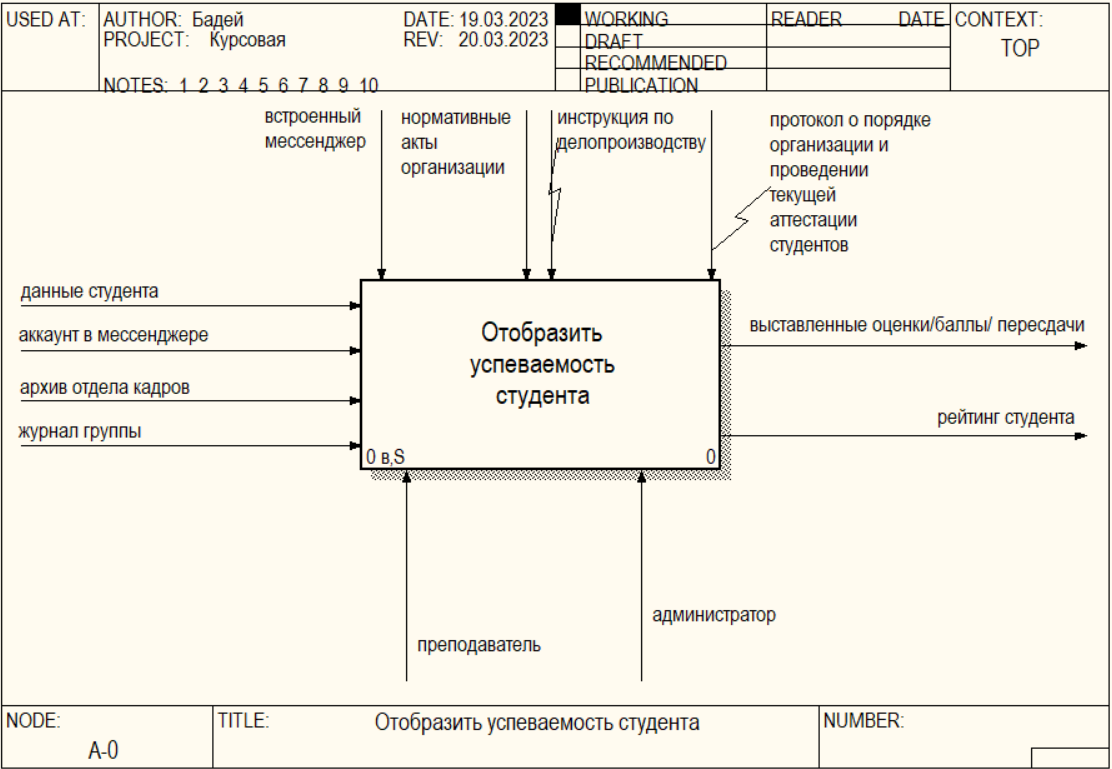


Рисунок 1.1 – Контекстная диаграмма модели А-0

На рисунке 1.2 представлена декомпозиция процесса отображения успеваемости студента. В данном случае работа делится на 4 блока: записать информацию о студенте, определить студента в группу, выставить оценки/баллы/пересдачи, сформировать рейтинг студента.

На схеме наглядно видно, что в процессе в основном принимает участие администратор: записать информацию о студенте, определить студента в группу, сформировать рейтинг студента.

В первый блок входят данные студента и архив отдела кадров, на выходе получаем информацию о студенте, блок выполняется администратором, руководствуясь инструкцией по делопроизводству и нормативными актами организации.

Во второй блок входят следующие данные: информация о студенте, аккаунт в мессенджере и журнал группы. На выходе получаем студента группы, блок так же выполняется администратором, который руководствуется в данном случае инструкцией по делопроизводству, встроенным мессенджером и нормативными актами.

В третий блок входит студент группы и журнал группы, на выходе получаем выставленные баллы/оценки/пересдачи. Блок выполняется преподавателем, руководствующимся протоколом о порядке организации и проведении текущей аттестации студентов, нормативными актами организации.

На входе четвертого блока получаем выставленные оценки/баллы/пересдачи, на выходе – рейтинг студента. Выполняется блок администратором, руководствуемся протоколом о порядке организации и проведении текущей аттестации студентов, нормативными актами организации.

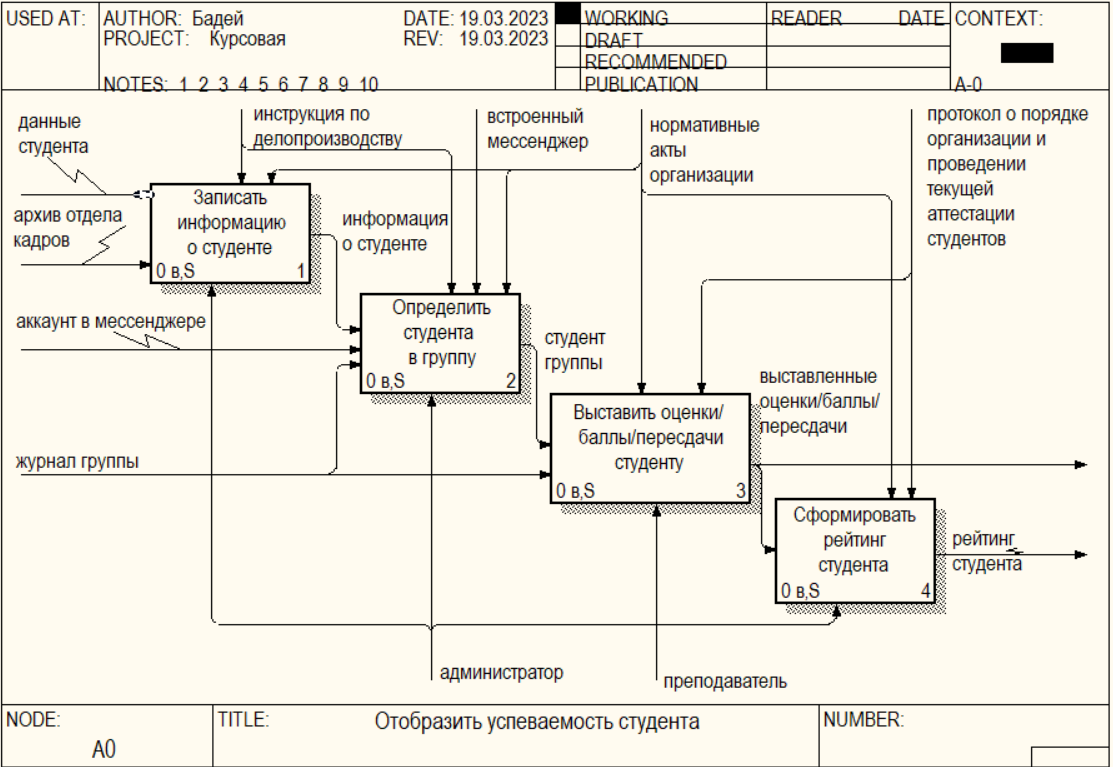


Рисунок 1.2 – Декомпозиция контекстной диаграммы

На рисунке 1.3 изображена декомпозиция процесса формирования рейтинга студента. Работа в данном случае так же делится на 4 блока: “вычислить среднюю оценку по всем предметам за текущий семестр”, “сравнить среднюю оценку студента со средней оценкой каждого студента потока”, “выделить место в рейтинговой таблице опираясь на результаты сравнения”, “внести студента в рейтинговую таблицу”. Администратор в данном случае управляет всеми процессами.

В первый блок входят выставленные оценки/баллы/пересдачи, на выходе получаем среднюю оценку. Руководствуемся протоколом о порядке организации и проведении текущей аттестации студентов, нормативными актами.

Во второй блок входит средняя оценка, на выходе получаем результат сравнения. Руководствуемся нормативными актами организации.

Входит в третий блок результат сравнения, выходит – место в таблице. Руководствуемся нормативными актами организации.

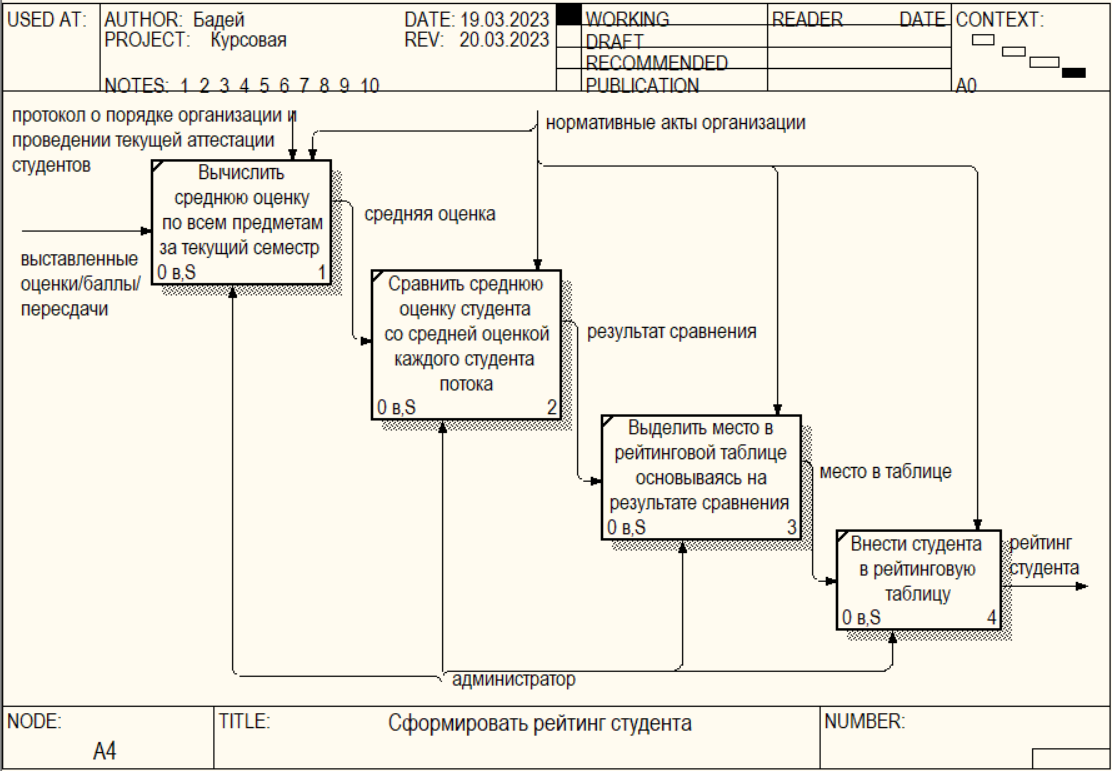


Рисунок 1.3 – Декомпозиция процесса «Сформировать рейтинг студента»

В четвертый блок входит место в таблице, выделенное под данного студента, и на выходе получаем рейтинг студента. Руководствуемся нормативными актами организации.

### **1.3 Разработка BPMN-модели основного процесса предметной области**

BPMN (Business Process Model and Notation) - это графический язык моделирования, который используется для описания бизнес-процессов и операций в компании [4].

BPMN-модели - это графические диаграммы, которые показывают бизнес-процессы внутри компании и их взаимодействие с внешними системами. Эти диаграммы могут быть использованы для описания как текущих, так и проектируемых бизнес-процессов. Моделирование процессов с помощью BPMN позволяет более эффективно управлять бизнес-процессами и обеспечивать их оптимизацию.

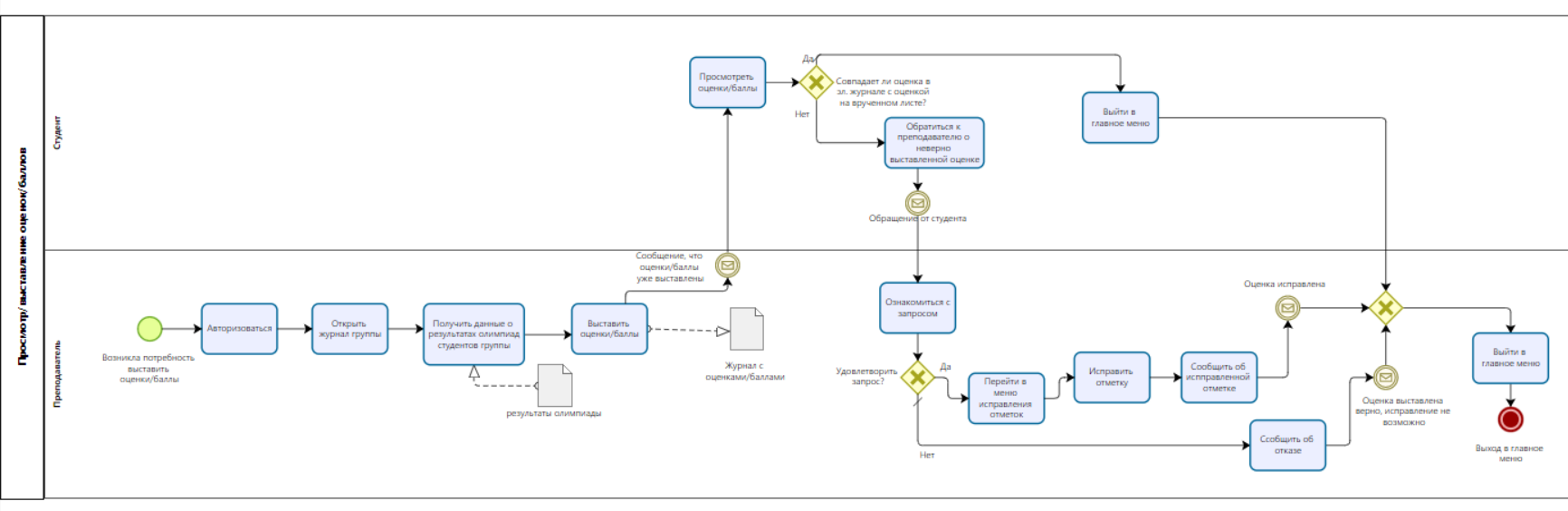


Рисунок 1.4 – Схема процесса «Просмотр и выставления оценок/баллов»

В общем и целом, BPMN-модели являются полезным инструментом для анализа, оптимизации и управления бизнес-процессами. Они позволяют описывать бизнес-процессы в форме графических диаграмм, которые могут быть использованы для понимания и улучшения производительности компании.

### **1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований.**

Разрабатываемое в ходе курсовой работы программное средство должно соответствовать определенным требованиям. Программное средство следует разработать на языке программирования C/C++ как консольное приложение. Парадигма программирования – процедурная. В готовом приложении должно быть разделение на пользователя и администратора с дальнейшим разделением их в правах. Модуль пользователя должен позволять просматривать данные о своих оценках, баллах и пересдачах. Каждая логически завершенная подзадача программы должна быть реализована в виде отдельной функции . Так же должна быть возможность осуществлять поиск данных и сортировку по нескольким параметрам. Помимо этого, необходимо предусмотреть запись данных в файл, обработку исключительных ситуаций (например, введенные пользователем данные не соответствуют формату поля, файл с записями для чтения не существует).

Разработанное программное средство должно обладать следующей инфраструктурой:

* интерфейс программного средства и данные представлять только на русском языке;
* программное средство должно запускаться без использования интегрированных средств разработки.

### **1.5 UML-модели представления программного средства и их описание**

Основными пользователями (актерами) реализовываемого программного средства являются администратор, преподаватель и студент. Для каждого из них предусмотрены свои определенные действия, причем некоторые действия администратора доступны преподавателю. Схема данной диаграммы можно увидеть на рисунке 1.5.

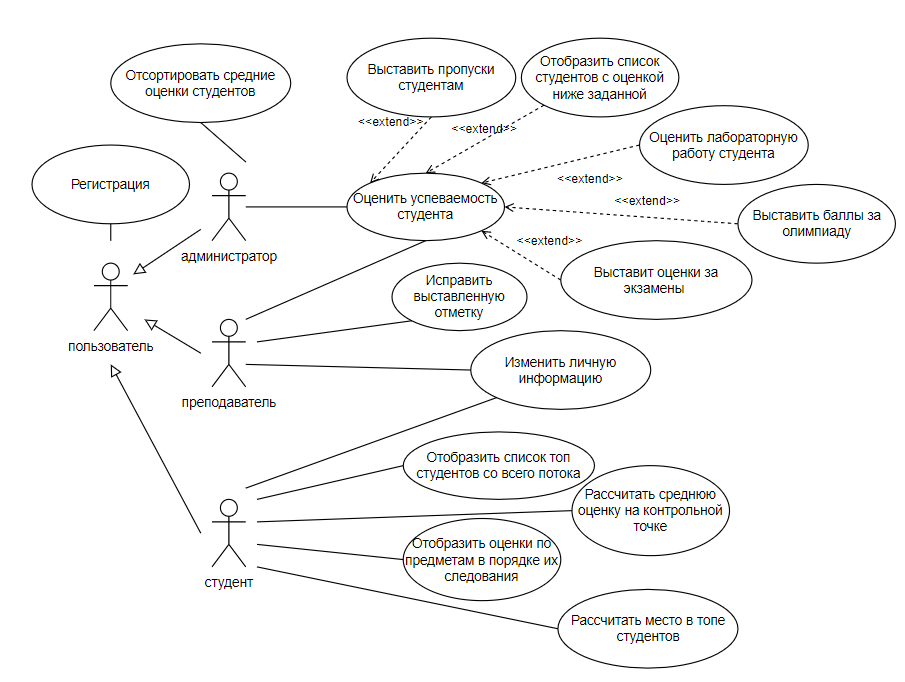


Рисунок 1.5 – UML – диаграмма вариантов использования

Диаграмма вариантов использования (Use Case Diagram) - это графический инструмент, который используется в разработке программного обеспечения для моделирования функциональных требований к системе с точки зрения ее пользователей или актеров. Диаграмма вариантов использования помогает определить функциональность, которую система должна предоставлять, а также идентифицировать актеров, которые будут использовать эту функциональность.

Из особенностей данной диаграммы можно выделить отношение обобщения для студента, преподавателя и администратора.

На начало работы программы они все являются пользователем, после регистрации каждому дается своя роль.

Диаграмма последовательности (Sequence diagram) - это графический инструмент, используемый в моделировании и проектировании программного обеспечения для описания взаимодействия между объектами или компонентами в системе в определенной последовательности [4].

Диаграмма последовательности используется для показа взаимодействия между объектами или компонентами в системе в определенном порядке с учетом времени. Она может помочь разработчикам программного обеспечения понять, как компоненты системы взаимодействуют друг с другом, и какие сообщения, вызовы методов и данные пересылаются между ними.

Диаграмма последовательности также может помочь в выявлении проблем в системе, таких как конфликты или задержки во времени, которые могут возникнуть в процессе взаимодействия.

UML-диаграмма последовательностей процесса авторизации продемонстрирована на рисунке 1.6.

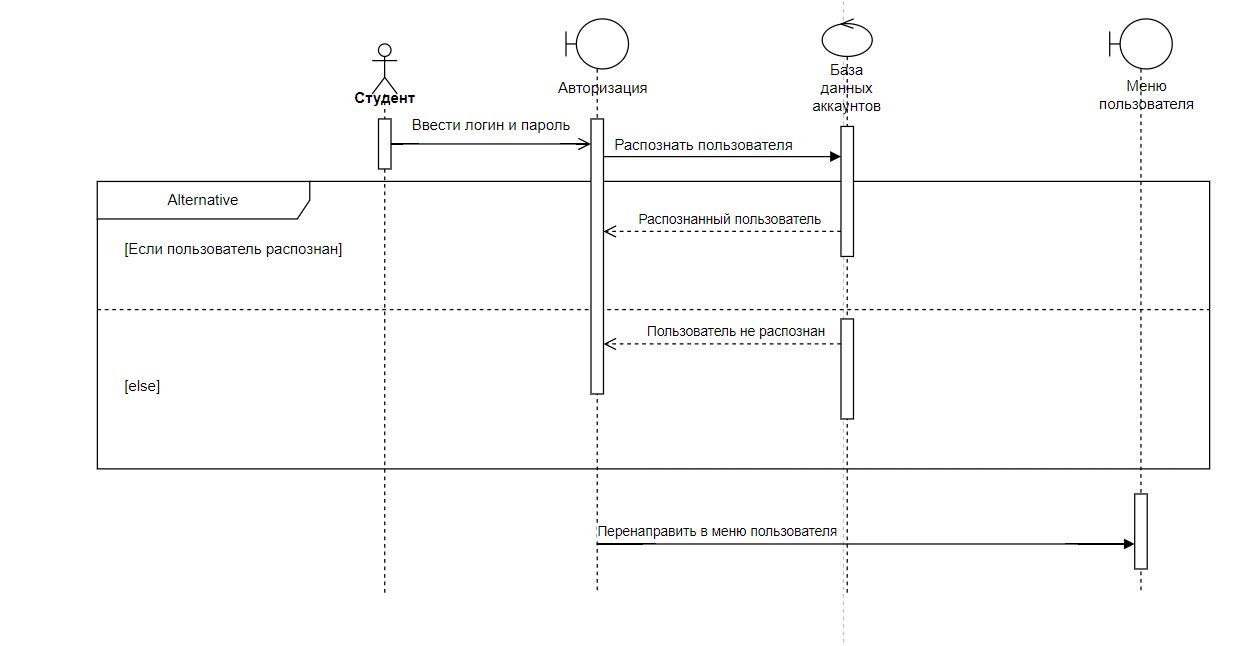


Рисунок 1.6 – UML-диаграмма последовательности

В общем и целом, диаграмма последовательности является полезным инструментом для анализа и проектирования систем, которые включают взаимодействие между несколькими объектами или компонентами, особенно в сложных системах с множеством компонентов, интерфейсов и взаимодействий.

Диаграмма компонентов (Component diagram) - это графический инструмент, используемый в разработке программного обеспечения для моделирования компонентов системы и их зависимостей, а также для описания архитектуры системы [4].

Диаграмма компонентов помогает разработчикам программного обеспечения понимать структуру системы и ее компонентов, их взаимосвязи и зависимости друг от друга. Она позволяет выделить компоненты, которые являются ключевыми для функционирования системы, а также определить их интерфейсы и взаимодействие друг с другом.

Кроме того, диаграмма компонентов позволяет определить зависимости компонентов от внешних библиотек или других систем. Данная диаграмма продемонстрирована на рисунке 1.7.

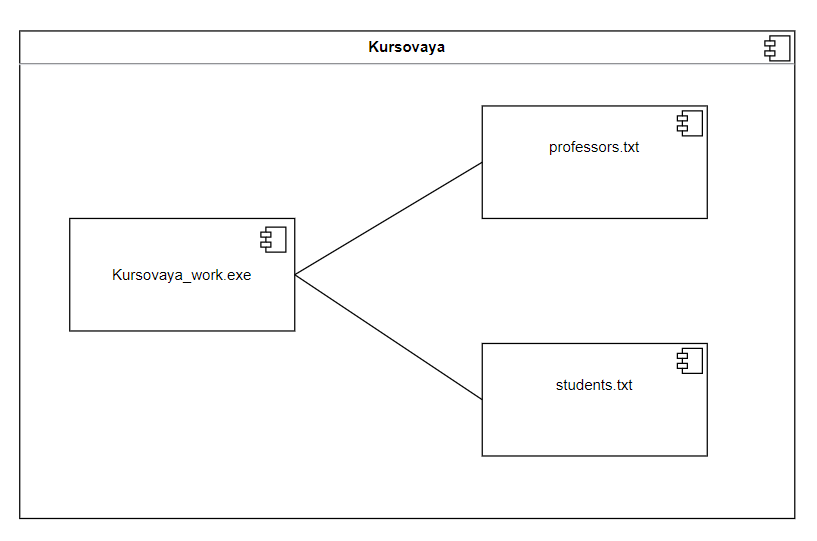


Рисунок 1.7 – UML-диаграмма компонентов

Диаграмма развертывания (Deployment diagram) - это графический инструмент, используемый в разработке программного обеспечения для моделирования физической структуры системы, включая аппаратное и программное обеспечение, а также их взаимосвязи и зависимости друг от друга [4].

Диаграмма развертывания помогает разработчикам программного обеспечения понимать, как система должна быть развернута на аппаратном и программном обеспечении, какие компоненты системы должны быть установлены на каких устройствах и как они должны взаимодействовать друг с другом. Она позволяет определить, какие устройства будут использоваться для выполнения различных функций системы, какие операционные системы будут использоваться, какие сетевые протоколы будут использоваться для обмена данными и т.д.

Диаграмма развертывания может быть использована для общения между разработчиками программного обеспечения и администраторами систем, чтобы улучшить понимание физической структуры системы, ее компонентов и их зависимостей друг от друга. Она также может помочь в выявлении потенциальных проблем в развертывании системы, таких как несовместимость компонентов или нарушение соответствия требованиям системы.

Диаграмма развертывания продемонстрирована на рисунке 1.8.

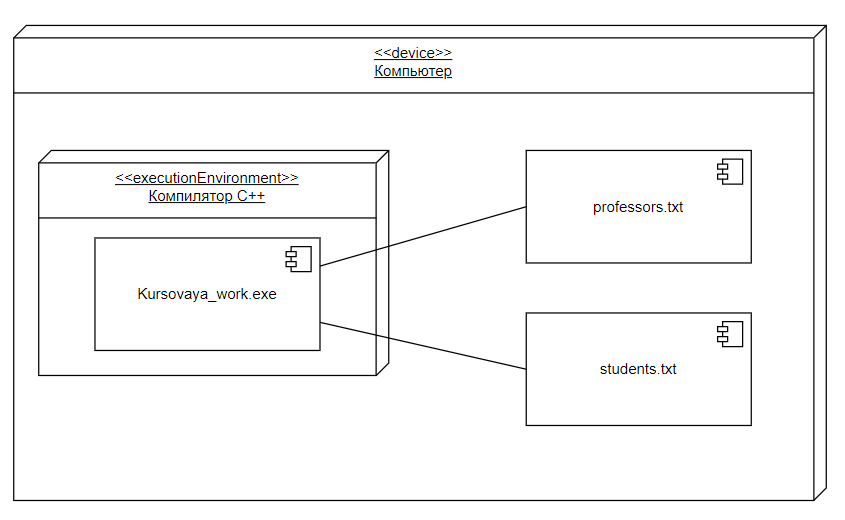


Рисунок 1.8 – UML-диаграмма развертывания

Диаграмма состояния (State diagram) - это графический инструмент, используемый в разработке программного обеспечения для моделирования поведения объекта или системы в различных состояниях и для описания переходов между состояниями [4].

Диаграмма состояния помогает разработчикам программного обеспечения понимать, как объект или система будет вести себя в разных условиях, и как она будет реагировать на различные внешние события.

Данная диаграмма может быть использована для общения между разработчиками программного обеспечения, чтобы улучшить понимание поведения объекта или системы в различных условиях, а также для документирования поведения системы. Она также может помочь в выявлении потенциальных проблем в поведении системы, таких как некорректная обработка входных данных или нарушение правил бизнес-логики.

Диаграмма состояния продемонстрирована на рисунке 1.9.

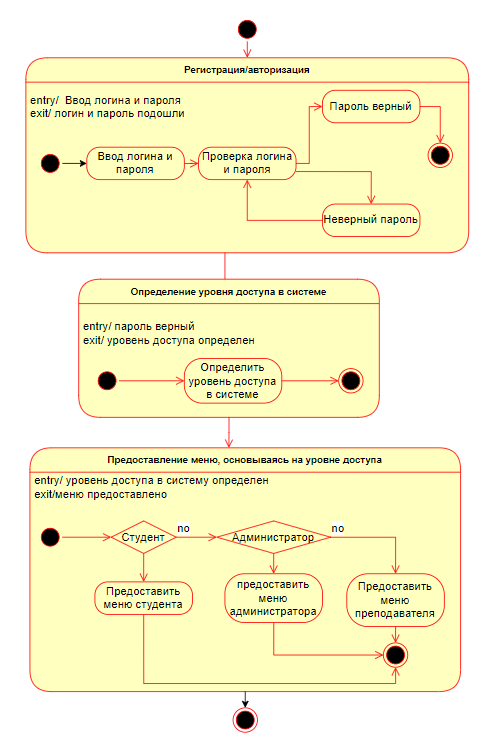


Рисунок 1.9 – UML-диаграмма состояния

В общем и целом, диаграмма состояния является полезным инструментом для анализа, проектирования и документирования поведения объекта или системы в различных условиях.

## 2 ПРОЕКТРИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

### **2.1 Постановка задачи**

По условию задания требуется разработать информационную систему учета успеваемости студента. Система должна предусматривать режимы ведения данных о студенте, отражающее все сведения о нем, хранящееся в электронном журнале. О каждом студенте будет известно следующее:

* фамилия имя отчество;
* номер группы;
* логин;
* оценки по определенному предмету.

Помимо данных о студенте, в ИС хранятся данные о преподавателях, а именно:

* фамилия имя отчество;
* логин;
* преподаваемый предмет.

Для информационной системы предусмотрены следующие ограничения:

* фамилия имя отчество должны начинаться с заглавных букв, не допускается использование заглавных букв в других местах;
* выставление оценок осуществимо в диапазоне от 1 до 10, другие значения не доступны;
* логин должен состоять из латинских букв, без использования некоторых специальных символов;
* пароль должен состоять из латинских букв, без использования некоторых специальных символов;

При выборе диапазона отметок необходимо ввести правую границу таким образом, чтобы она была больше 2 или больше значения правой границы, а также меньше или равна 10. Левая граница должна быть не меньше 1, а также меньше 10.

При работе с системой преподаватель должен иметь возможность выставлять оценки по экзаменам, олимпиадам и лабораторным работам, а также выставлять пропуски.

Студент же в свою очередь должен иметь возможность просматривать оценки, пропуски по определенным предметам.

### **2.2 Разработка модульной структуры**

Разработка модульной структуры является важным этапом процесса создания программного продукта. Модульная структура позволяет разбить программу на более мелкие компоненты, называемые модулями, которые могут быть разработаны и отлажены независимо друг от друга.

Преимущества модульной структуры включают более простое тестирование и отладку, легкую поддержку и расширение программы, а также возможность повторного использования кода. Кроме того, модульная структура позволяет более эффективно использовать ресурсы компьютера и обеспечивает более высокую надежность и безопасность программного продукта.

Для разработки системы был выбран метод восходящей разработки. Модульная структура представлена на рисунке 2.1.

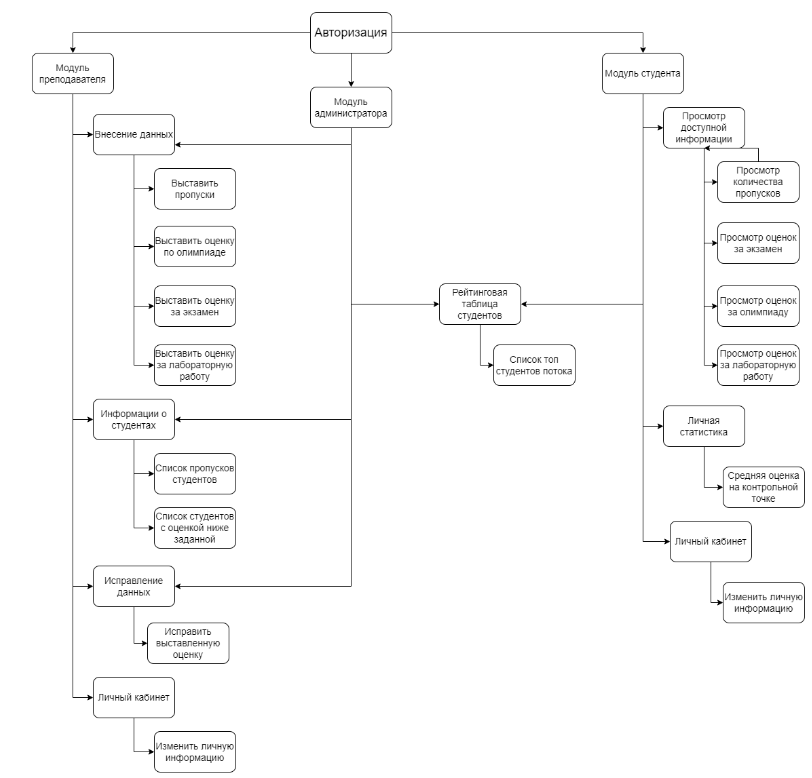


Рисунок 2.1 – Модульная структура программы

При разработке модульной структуры необходимо определить функциональные блоки программы и их взаимодействие. Каждый модуль должен иметь четко определенные интерфейсы взаимодействия с другими модулями, чтобы обеспечить их взаимную совместимость и безопасность.

### **2.3 Выбор способа организации данных**

В ходе анализа требований к курсовой работе было принято решение хранить и взаимодействовать с хранимыми данными через структуры.

Существует несколько причин, почему использование структур для хранения данных в C++ может быть полезным:

* упорядоченность данных: Структуры позволяют объединить несколько переменных разных типов в одну единую структуру, что позволяет упорядочить данные и облегчает их доступ.
* логическая группировка: Структуры также могут использоваться для логической группировки переменных, что может быть полезно при написании кода, когда некоторые переменные имеют связь друг с другом.
* сокрытие информации: Структуры могут использоваться для сокрытия информации, объединяя несколько переменных в одну структуру и делая доступ к этой информации более удобным и безопасным.
* удобство передачи данных: Структуры могут передаваться как единый параметр в функции, что упрощает передачу нескольких переменных разных типов.
* экономия памяти: Использование структур может помочь экономить память, поскольку несколько переменных могут быть объединены в одну структуру.

На рисунке 2.2 изображена структура Professor, которая хранит в себе информацию о Ф.И.О., преподаваемому предмету, логине и пароле преподавателя.

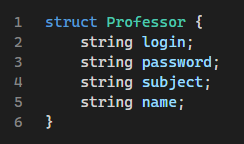


Рисунок 2.2 – Структура Professor

В целом, использование структур позволяет более удобно и эффективно работать с группами связанных данных, делая код более организованным, читаемым и безопасным.

С другой стороны, использование классов выглядело бы не менее эффективно, т.к. можно было бы создавать поля с разными модификаторами доступа (private, protected, public). Также можно было бы организовать наследование классов, где основной класс был пользователем и содержал в себе поля Ф.И.О., логин, пароль и , возможно, некоторые методы.

Но в процессе разработки программы пришлось отказаться от данной идеи в пользу простоты использования структур, как структуры хранения данных.

На рисунке 2.3 изображена структура Users, в которой хранится информация о студенте.

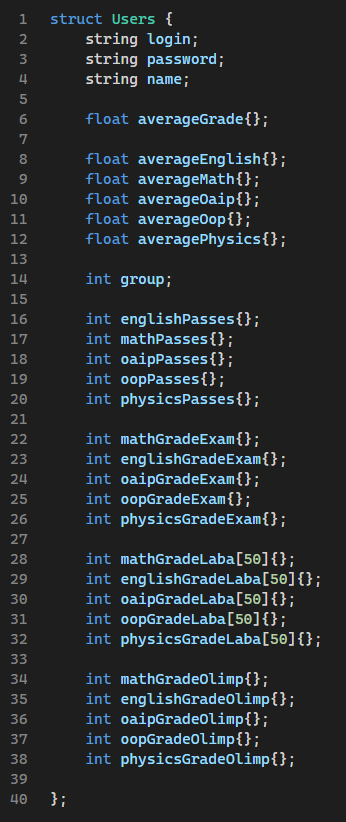


Рисунок 2.3 – Структура Users, содержащая информацию о студенте

Как видно из рисунка, структура хранит в себе следующую информацию:

* Логин;
* Пароль;
* Ф.И.О.;
* Средняя отметка;
* Средняя отметка по каждому предмету;
* Номер группы;
* Пропуски по каждому предметам;
* Оценка за экзамен по каждому предмету;
* Оценка за лабораторные работы по каждому предмету;
* Оценка за олимпиаду по каждому предмету.

В качестве способа объединения входных данных принято решение использовать статические массивы, т.к. выделение динамической памяти под каждый новый созданный объект будет выглядеть лишним для решения поставленной задачи, в виду того, что можно ограничиться выделением статической памяти под массив студентов и преподавателей.

С другой стороны, выделение динамической памяти выглядело бы более разумно и правильно, если бы данный проект был глобальнее.

### **2.4 Разработка перечня пользовательских функций программы**

void recordLabaGrades(); // запись данных студента в файл

void recordProfessors(); // запись данных преподавателя в файл

bool readProfessorFile(); // чтение данных преподавателя из файла

bool readStudentFile(); // чтение данных студента из файла

void MainAdminMenu(); // меню администратора

void rateExam(); // выставить оценки за экзамен

void rateOlimp(); // выставить оценки за олимпиаду

void rateLaba(); // выставить оценку за лабораторную работу

void gradesList(); // список оценок

void putUpPasses();

// выставить пропуски

void outputPasses();

// просмотреть пропуски

void MainStudentMenu(int count); // меню студента

void personalStudentsCabinet(int count); // личный кабинет студента

void MainProfessorMenu(int count); // меню преподавателя

void personalProfessorsCabinet(int count); // личный кабинет преподавателя

void changeProfessorProfileInfo(int count); // изменить личную информацию

void box(); // шапка таблицы

void olimpGrades(int count); // оценки по олимпиаде

void outputAverageGrade(int count); // просмотр средней оценки

void examGrades(int count);

// просмотр оценки за экзамен

void subjectsGrades(int count); // просмотр всех отметок

void studentPasses(int count); // пропуски студента

void outputPasses(int count); // просмотреть пропуски

void infoTablePasses(); // просмотреть пропуски в таблице

void Passes(int passes, int group, string name, string login, int y, int i); // пропуски

void putUpPasses(int count); // выставить пропуски

int checkNam(int num, int count, int y); // проверка

int checkNum(int num, int count, int i); // проверка

void gradesList(int count); // просмотреть список оценок

void englishGradesList(); // список оценок по английскому

void mathGradesList(); // список оценок по математке

void oaipGradesList(); // список оценок по оаип

void oopGradesList(); // список оценок по ооп

void physicsGradesList(); // список оценок по физике

int choiceTable(int y); // выбор студента

void rateExam(int count); // выставить оценку за экзамен

void rate\_Exam(int count, int\* grade);

void rateOlimp(int count);// выставить оценку по олимпиаде

void rate\_Olimp(int count, int \*grade);

void rateLaba(int count); // выставить оценку по лабораторной работе

void rate\_Laba(int count, int\* grade);

void RegistrationMenu(); // главное меню

string pressOutFunc(string str); // расшифровка пароля

int Registration(int i); // меню регистрации

string passwordEntering(); // проверка

void regStudent(string login, string password); // меню регистрации студента

bool checkName(string name); // проверка

void regProfessor(string login, string password); // меню регистрации препода

void recordData(); // шапка таблицы

bool CheckUsers(string login, string password); // проверка

string PressmarkFunc(string str); // шифровка пароля

void Autorization(); // меню авторизации

bool checkLogin(string login); // проверка

bool checkPassword(string str); // проверка

void AutorizationKursor(string login, string password); // проверка

bool CheckAutorization(string login, string password); // проверка

void GoToXY(short x, short y); // перемещение курсора

int CheckNumber(int text); // проверка

void Loading(); // загрузка

int checkGroup(int group); // проверка

void topStudents(); // вывод списка топ студентов

void averageGrade(); // расчёт средней оценки студента

void pulsar(); // для продолжения нажмите любую клавишу

bool studentsCount(); // проверка

### **2.5 Разработка схем алгоритмов работы программы**

Схема алгоритма призвана упростить понимание работы того или иного участка программного кода, а также наглядно продемонстрировать операции, происходящие в рассматриваемом участке кода.

Схемы алгоритмов помогают разработчикам лучше понять логику работы программы. Они представляют собой визуальное представление шагов и процессов, которые должна выполнить программа для достижения определенной цели. Разработчики могут использовать схемы алгоритмов для анализа и оптимизации логики программы, выявления потенциальных проблем и улучшения ее эффективности.

Схемы алгоритмов служат средством коммуникации между разработчиками и другими заинтересованными сторонами. Они помогают описать и объяснить логику программы в понятной форме, что особенно полезно при работе в команде или при взаимодействии с заказчиками и пользователями. Схемы алгоритмов могут быть использованы для обсуждения и уточнения требований к программе, выявления проблем и нахождения согласия по поводу функциональности.

Блок-схема алгоритма работы функции mainMenu() приведена ниже на рисунке 2.4.

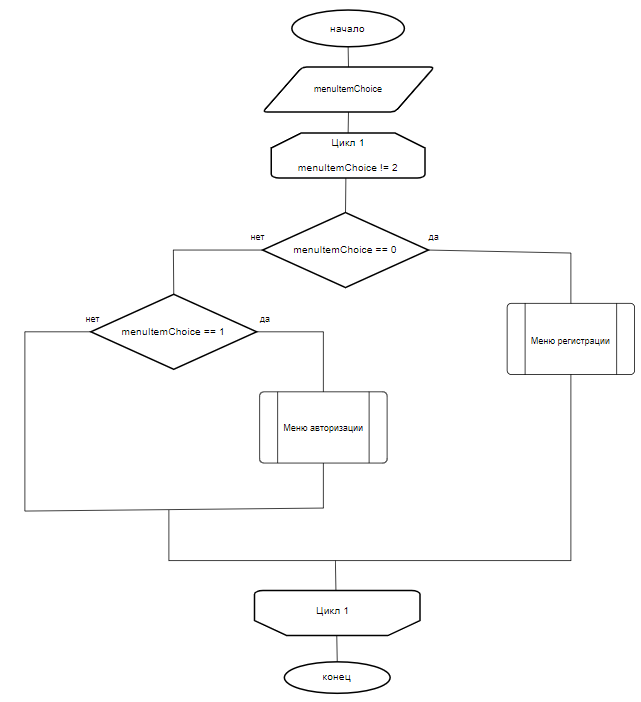


Рисунок 2.4 – Блок-схема работы алгоритма функции mainMenu

Данная функция является главным меню программы. Здесь возможно авторизоваться, зарегистрироваться и выйти из программы.

В этой же функции происходит считывание из файла всей необходимой информации о студенте и преподавателе.

Листинг кода:

void mainMenu()

{

setlocale(LC\_ALL, "Rus"); // подкл рус яз

system("chcp 1251"); // два раза подк рус яз потому что при записи в файл кракозябра без систем

system("cls");

if (!readStudentFile() || !readProfessorFile())

{ // считываем инфу из файла и записываем в структуру

system("cls");

exit(0);

}

string Menu[] = { "Регистрация", "Авторизация", "Выход"};

int active\_menu = 0; // будем бегать по меню изменяя эту переменную

char ch;

while (1)

{

int x = 77, y = 20;

GoToXY(67, y - 1);

SetConsoleTextAttribute(hStdOut, 7);

cout << "————————— Главное меню —————————" << endl;

for (int i = 0; i < size(Menu); i++)

{

if (i == active\_menu) SetConsoleTextAttribute(hStdOut, 6);

else SetConsoleTextAttribute(hStdOut, 7);

GoToXY(x, ++y);

cout << Menu[i] << endl;

}

GoToXY(67, y + 2);

SetConsoleTextAttribute(hStdOut, 7);

cout << "————————————————————————————————" << endl;

ch = \_getch(); //код нажатой клавиши

if (ch == -32) ch = \_getch(); // Отлавливаем стрелочки (2 байтовые)

switch (ch)

{

case ESCAPE:

{

SetConsoleTextAttribute(hStdOut, 7);

recordLabaGrades();

exit(0);

}

case UP:

if (active\_menu > 0)

{

--active\_menu;

}

else

{

active\_menu = size(Menu) - 1;

}

break;

case DOWN:

if (active\_menu < size(Menu) - 1)

{

++active\_menu;

}

else

{

active\_menu = 0;

}

break;

case SPACE:

switch (active\_menu)

{

case 0:

{

system("CLS");

int i = 0;

i = Registration(i);

if (i == 0) MainStudentMenu(CountOfUsers - 1);

else if(i == 1) MainProfessorMenu(CountOfProfessors - 1);

system("CLS");

break;

}

case 1:

{

system("CLS");

Autorization();

system("CLS");

break;

}

case 2:

}

SetConsoleTextAttribute(hStdOut, 7);

recordLabaGrades();

recordProfessors();

exit(0);

}

}

break;

default:

break;

}

}

}

На рисунке 2.5 приведена блок-схема данной функции.

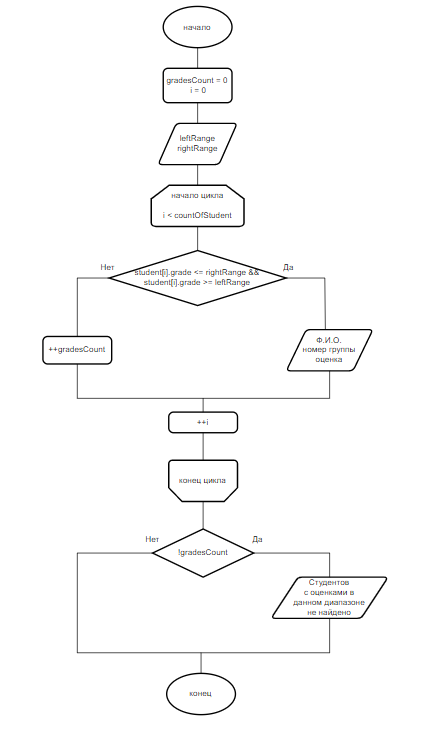


Рисунок 2.5 – Блок-схема работы алгоритма функции gradesList

При вводе границ отметок, необходимо руководствоваться ограничениями, приведенными мной выше, иначе пользователь будет перенаправлен обратно в меню преподавателя.

Также рассмотрим блок-схему функции checkGroup.

Листинг кода:

int checkGroup(int group) {

while (!(cin >> group) || group > 999999 || group < 100000) {

cin.clear();

cin.ignore(1024, '\n');

GoToXY(70, 22);

SetConsoleTextAttribute(hStdOut, 4);

cout << "Номер группы должен состоять из 6 цифр!";

SetConsoleTextAttribute(hStdOut, 7);

\_getch();

GoToXY(70, 22);

cout << " ";

GoToXY(70, 22);

cout << "Введите номер своей группы: ";

}

return group;

}

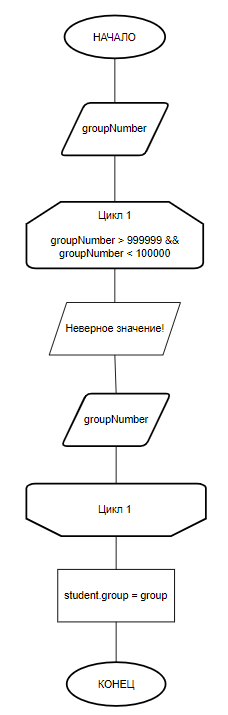


Рисунок 2.6 – Блок-схема работы функции checkGroup

Функция проверяет корректность ввода номера группы студента(рисунок 2.6). При некорректном вводе номера группы пользователь обязан снова ввести его

## 3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

В программе реализованы следующие исключения и проверки:

* проверка на Ф.И.О.;
* проверка логина;
* проверка пароля;
* проверка выставленной отметки;
* проверка выставленного пропуска;
* проверка границ выбора отметок;
* проверка на совпадение логинов.

Рассмотрим реализацию некоторых приведенных проверок в программе (рисунок 3.1 – 3.4).

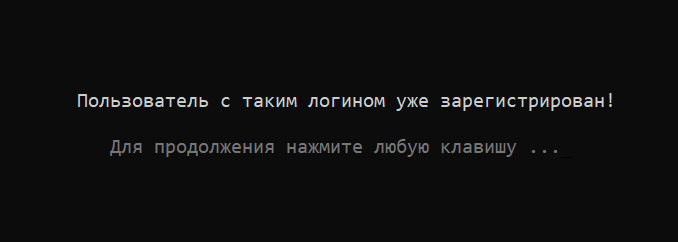


Рисунок 3.1 – Проверка на совпадение логинов

Регистрация под логином, схожим с логином другого пользователя, невозможно, дабы при поиске данные логины не путались.

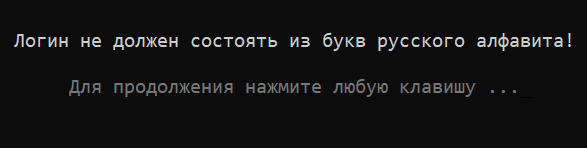


Рисунок 3.2 – Проверка корректности ввода логина

Использование русских букв при вводе пароля или логина недопустима, чтобы снизить нагрузку на систему.

Также было исключено использование некоторых специальных символов по типу слэша, обратного слэша, знака процента, доллара и похожих нестандартных для записи пароля символов.

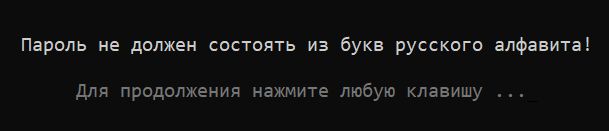


Рисунок 3.3 – Проверка корректности ввода пароля

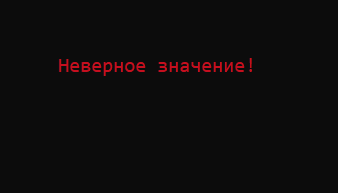


Рисунок 3.4 – Проверка ввода отметки

Оставшиеся функции-проверки имеют схожие сообщения при неверном вводе того или иного параметра, поэтому на этом наше рассмотрение функций-проверок окончен.

## 4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ПРИМЕР

### **4.1 Авторизация**

Теперь давайте рассмотрим поведение программы, если бы пользователь захотел авторизоваться в качестве преподавателя (рисунок 4.1). До ввода логина и пароля следуем тем же инструкциям и при выборе “Зарегистрироваться” или “Ввести ключ” выбираем второй вариант и вводим ключ регистрации преподавателя “professor123”. Данный ключ в программе является константным.

Как видно из рисунка, нам необходимо ввести ключ, упомянутый текстом выше. При неверном вводе ключа на экране появляется соответствующее сообщение и пользователь перенаправляется в главное меню.

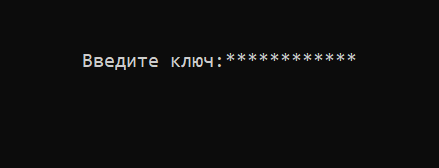


Рисунок 4.1 – Ввод ключа регистрации преподавателя

Далее необходимо ввести Ф.И.О. (рисунок 4.2) с такими же проверками, как и при регистрации студента.

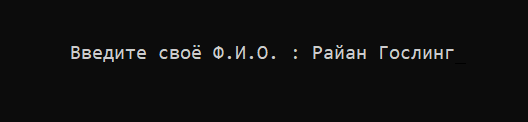


Рисунок 4.2 – Ввод Ф.И.О.

Далее преподавателю предоставляется на выбор преподаваемый предмет (рисунок 4.3), выбрать можно только единственный предмет из приведенных на рисунке.

На выбор было предоставлен список основных предметов, изучающихся в текущем семестре, а именно:

* физика;
* английский язык;
* математический анализ;
* основы алгоритмизации и программирования;
* объектно-ориентированное программирование и проектирование.

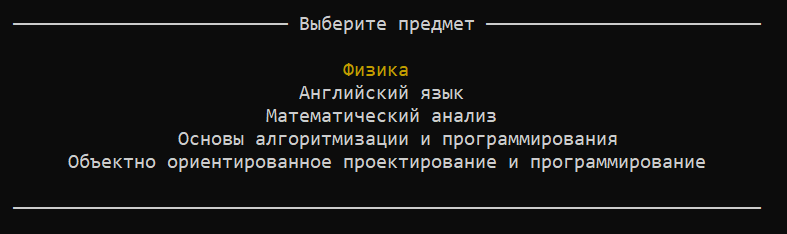


Рисунок 4.3 – Меню выбора предмета

Преподавателю необходимо выбрать один предмет из приведенных.

Для наглядности выберем математический анализ и завершим этап регистрации в качестве преподавателя.

После успешной регистрации нам перенаправляют в меню преподавателя. Меню можно рассмотреть на рисунке 4.4.

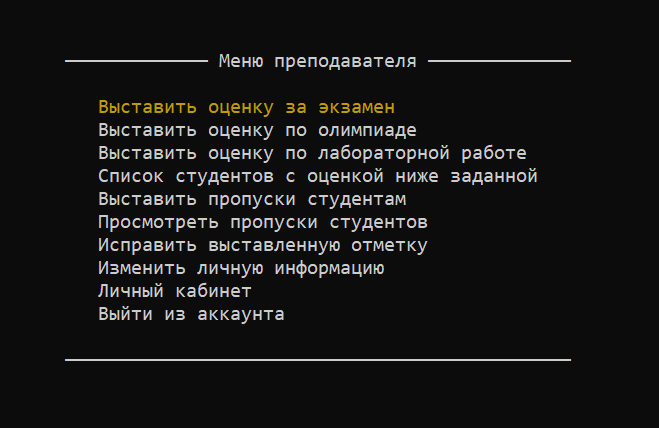


Рисунок 4.4– Меню преподавателя

Модуль студента был рассмотрен нами ранее в предыдущей главе, поэтому возвращаться к модулю студента не совсем цедесообразно.

### **4.2 Модуль администратора**

Рассмотрим модуль администратора. Администратор в данной программе является константой, т.е. логин и пароль для администратора уже заданы. Зарегистрироваться под логином администратора невозможно.

Чтобы получить доступ к меню администратора необходимо авторизоваться под логином администратора. Для этого в главном меню выбираем пункт “Авторизоваться”. Далее вводим логин и пароль администратора: admin и admin123 соответственно (рисунок 4.5).

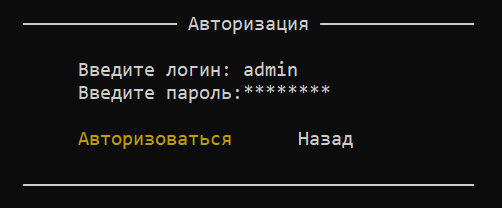


Рисунок 4.5– Авторизация администратора

Авторизовываемся и переходим в меню администратора, показанное на рисунке 4.6.

Ф.И.О. и прочую информацию вводить не благоразумно, т.к. роль администратора предполагает поддержание корректной работы системы.

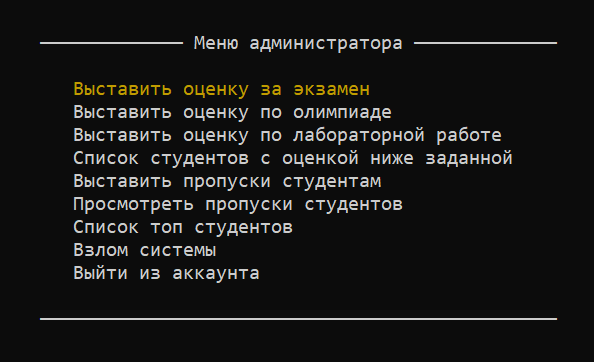


Рисунок 4.6 – Меню администратора

Роль администратора в данной программе схожа с ролью преподавателя, но все же является более обширной и развернутой. В общем и целом, администратор - это пользователь программы, который имеет полный доступ к ее функционалу и управляет ее работой; специалист, который управляет установкой, настройкой и обслуживанием приложений на серверах или в облачной среде.

Администратор отвечает за установку и настройку приложений, контроль за их работой и обновлением, настройку политик безопасности и доступа к данным, а также за резервное копирование и восстановление данных в случае необходимости.

Роль администратора в программе может быть необходима по следующим причинам:

1. Управление правами доступа: администратор может назначать различные уровни доступа для пользователей программы в зависимости от их роли и задач. Например, ученики могут иметь доступ только к своим учебным материалам, а учителя могут иметь право просматривать и оценивать работы учеников.
2. Управление данными: администратор может создавать, редактировать и удалять данные в программе. Например, он может добавлять новых пользователей, создавать учебные материалы, изменять настройки программы и т.д.
3. Решение проблем: администратор может помогать пользователям решать возникающие проблемы в работе программы. Например, он может помочь восстановить утерянный пароль, разобраться в технических проблемах и т.д.
4. Обеспечение безопасности: администратор может устанавливать и поддерживать меры безопасности программы, такие как шифрование данных, установка ограничений доступа и т.д.

Таким образом, администратор необходим для эффективной работы программы и обеспечения ее безопасности, управления данными и разрешения проблем, возникающих у пользователей.

### **4.3 Модуль пользователя**

Рассмотрим модуль пользователя, а именно студента. При регистрации студенту необходимо ввести дополнительную информацию, еоторая приводилось выше при рассмотрении 3 главы.

После ввода необходимой информации студент попадает в меню студента, где ему предоставлен полный спектр возможностей, продемонстрированных на рисунке 4.7.

Студенту предоставлены следующие пункты меню:

* вывести список топ студентов;
* просмотреть оценки за экзамен;
* просмотреть оценки по предметам;
* просмотреть количество пропусков;
* просмотреть баллы по олимпиадам;
* средняя оценка на контрольной точке;
* изменить личную информацию;
* личный кабинет.

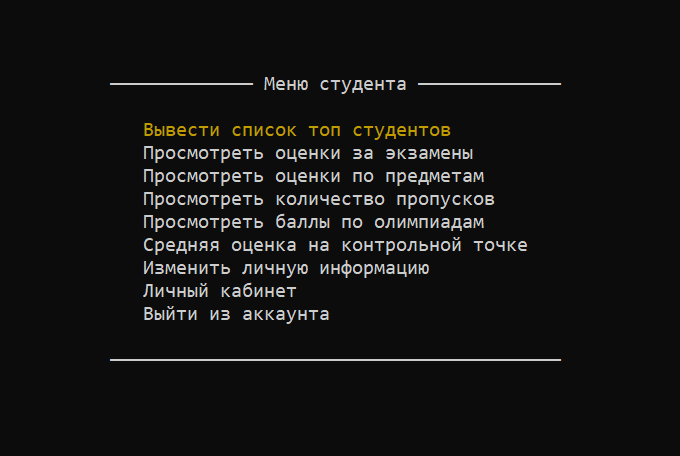


Рисунок 4.7 – Меню студента

Каждый из приведенных пунктов меню несет в себе ключ для организации работы студента, осведомленность о своих текущих отметках по определенным предметам, о количестве пропусков по каждому предмету. У студента есть возможность узнать свое место в рейтинговой таблице, свой средний на контрольной точке.

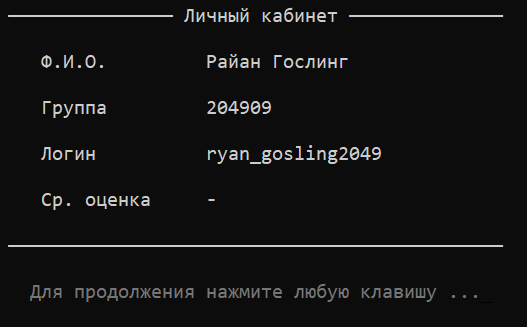


Рисунок 4.8 – Личный кабинет студента

В меню студента приведен весь необходимый функционал для работы с системой, для подчеркивания нужных сведений о своей успеваемости.

В личном кабинете студента представлена вся основная информация об аккаунте студента: логин, Ф.И.О., номер группы и средняя отметка(рисунок 4.8).

Также студенту предоставлена возможность вносить изменения о своих персональных данных, будь то логин, пароль, Ф.И.О. или номер группы. Это прибавляет некоторой пластичности при работе с данными, т.к. студент может внести неверные данные в начале работы с программой (рисунок 4.9).

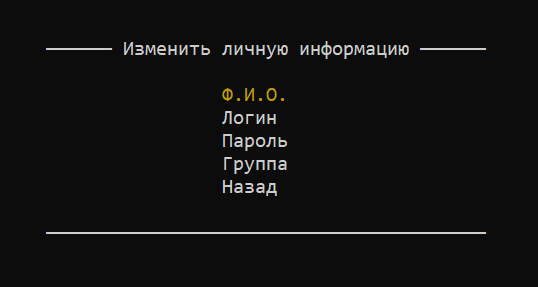


Рисунок 4.9 – Меню изменения личной информации студента

При изменении пароля, система будет запрашивать текущий пароль студента для проверки дополнительной безопасности. Новый пароль необходимо будет ввести дважды, чтобы удостовериться в надежности созданного пароля, а также, чтобы студент был точно уверен в том, что его новый пароль соответствует нужным требованиям и условиям.

Меню содержит все необходимое, чтобы студент смог организовывать себя и свой распорядок рабочего дня таким образом, чтобы достичь максимальных результатов при минимальном затрате сил.

## ЗАКЛЮЧЕНИЕ

В процессе разработки программного средства была изучена предметная область проекта, а также создан быстрый и удобный доступ к имеющейся информации о пациентах, их редактированию и просмотру; было разработано программное средство для учета успеваемости студентов, которое предназначено для автоматизации процесса ведения журнала успеваемости и оценок студентов.

Были проведены анализ требований и проектирование архитектуры системы, разработаны необходимые модули и функционал, произведено тестирование и отладка программного средства..

Была организована целостность структур данных посредством использования текстовых файлов для их хранения, а также обеспечена защита данных посредством авторизации пользователя в программе, а также шифрования данных авторизации.

Разработанные пользовательские функции позволяют эффективно работать с имеющимися данными посредством их ввода, просмотра. Редактирования удаления, поиска, сортировки и других операций.

Функциональная схема работы программы представлена в пояснительной записке в виде алгоритмов некоторых меню и функций.

Программное средство имеет понятный графический интерфейс, позволяющий с минимальными навыками использования персонального компьютера использовать преимущества хранения информации в цифровом виде. Таким образом, система готова к эксплуатации. Она может обеспечить пользователю поступление необходимой информации, а также облегчить получение статистических наблюдений.

Для понимания логики программного средства были разработаны модели бизнес-процессов предметной области в нотациях IDEF0 и BPMN, проектные решения программного средства на основе языка UML.

Были соблюдены все необходимые требования, описанные в бланке заданий, содержание пояснительной записки (перечень подлежащих разработке вопросов) было выполнено с учетом требований и ГОСТов.

Таким образом, разработанное программное средство представляет собой эффективный инструмент для автоматизации учета успеваемости студентов и может быть использовано в образовательных учреждениях различных уровней.

## Список использованных источников

[1] Шилдт, Г. С++. Полное руководство / Г. Шилдт. – М. : Вильямс,

2019. – 800 с.

[2] Цукахова О.А. Методология и инструментарий моделирования бизнес-процессов: учебное пособие-СПб.: Университет ИТМО, 2015. – 100с.

[3] Шилдт, Г. С++: базовый курс / Г. Шилдт. – 3-е издание. – М. :

Диалектика-Вильямс, 2018. – 624 с.

[4] Habr [Электронный ресурс]. – Режим доступа: <http://habr.com/ru/post/508710/>

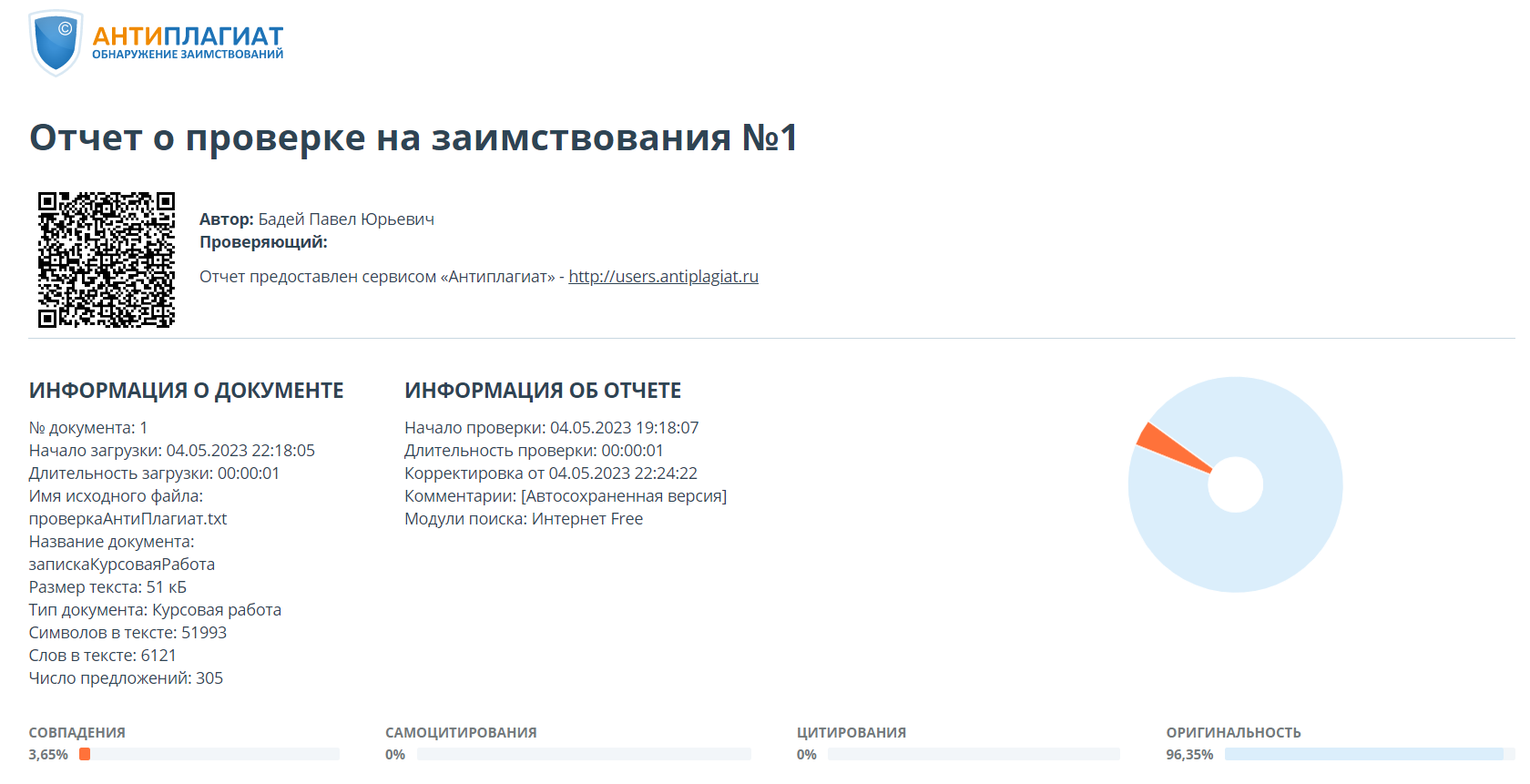
[5] Вендров А.М. В29 Проектирование программного обеспечения экономических информационных систем: Учебник. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2006. – 544с.

[6] Куклина И.Г., Информационные технологии и платформы разработки информационных систем: Куклина И.Г. – 2018. – 283с.

[7] Ларман, Крэг, Применение UML 2.0 и шаблонов проектирования. Практическое руководство. 3-е издание / Крэг Ларман. – СПб.: Вильямс, 2019-736с.

[8] Прата, С. Язык программирования С++. Лекции и упражнения / С.Прата. – М.:ООО “Издательский дом “Вильямс”, 2016. – 1248с.

## ПРИЛОЖЕНИЕ А (обязательное) Проверка на заимствование



## ПРИЛОЖЕНИЕ Б (обязательное) Листинг кода

void MainAdminMenu() {

system("cls");

//добавить список топ студентов и пентагон

string Menu[] = { "Выставить оценку за экзамен", "Выставить оценку по олимпиаде", "Выставить оценку по лабораторной работе", "Список студентов с оценкой ниже заданной", "Выставить пропуски студентам", "Просмотреть пропуски студентов", "Список топ студентов", "Взлом системы", "Выйти из аккаунта"};

int active\_menu = 0;

char ch;

while (1)

{

int x = 65, y = 20;

GoToXY(62, y - 1);

SetConsoleTextAttribute(hStdOut, 7);

cout << "————————————— Меню администратора —————————————" << endl;

for (int i = 0; i < size(Menu); i++)

{

if (i == active\_menu) SetConsoleTextAttribute(hStdOut, 6);

else SetConsoleTextAttribute(hStdOut, 7);

GoToXY(x, ++y);

cout << Menu[i] << endl;

}

GoToXY(62, y + 2);

SetConsoleTextAttribute(hStdOut, 7);

cout << "———————————————————————————————————————————————" << endl;

ch = \_getch(); //код нажатой клавиши

if (ch == -32) ch = \_getch(); // Отлавливаем стрелочки (2 байтовые)

switch (ch) {

case ESCAPE:

return;

case UP:

if (active\_menu > 0) {

--active\_menu;

}

else {

active\_menu = size(Menu) - 1;

}

break;

case DOWN:

if (active\_menu < size(Menu) - 1) {

++active\_menu;

}

else {

active\_menu = 0;

}

break;

case SPACE:

switch (active\_menu)

{

case 0:

if (studentsCount()) break;

rateExam();

break;

case 1:

if (studentsCount()) break;

rateOlimp();

break;

case 2:

if (studentsCount()) break;

rateLaba();

break;

case 3:

if (studentsCount()) break;

gradesList();

break;

case 4:

if (studentsCount()) break;

putUpPasses(); //пропуски

break;

case 5:

if (studentsCount()) break;

outputPasses();

break;

case 6:

if (studentsCount()) break;

topStudents();

break;

case 7: Pentagon();

break;

case 8:

system("cls");

return;

}

break;

default:

break;

}

}

}

//выставить оценки по экзамену

void rateExam() {

int y = 5;

system("cls");

GoToXY(70, y);

cout << "Выберите студента:";

y = choiceTable(y);

int i = 0;

for (i; i < CountOfUsers; ++i) {

y += 2;

GoToXY(57, y);

cout << i + 1;

GoToXY(60, y);

cout << " | " << clients[i].group;

GoToXY(70, y);

cout << "| " << clients[i].name;

GoToXY(100, y);

cout << "| " << clients[i].login;

GoToXY(55, y + 1);

cout << "——————————————————————————————————————————————————————————————";

}

int choice{};

choice = checkNum(choice, CountOfUsers, i);

--choice;

system("cls");

string Menu[] = { "ОАиП", "ООПиП", "Физика", "Математика", "Английский язык", "Назад"};

int active\_menu = 0;

char ch;

while (1)

{

int x = 77, y = 20;

GoToXY(72, y - 1);

SetConsoleTextAttribute(hStdOut, 7);

cout << "—————— Выберите предмет ——————" << endl;

for (int i = 0; i < size(Menu); i++)

{

if (i == active\_menu) SetConsoleTextAttribute(hStdOut, 6);

else SetConsoleTextAttribute(hStdOut, 7);

if (i == 0) GoToXY(85, ++y);

else if (i == 1) GoToXY(84, ++y);

else if (i == 2) GoToXY(83, ++y);

else if (i == 3) GoToXY(82, ++y);

else if (i == 4) GoToXY(x + 3, ++y);

else GoToXY(84, ++y);

cout << Menu[i] << endl;

}

GoToXY(72, y + 2);

SetConsoleTextAttribute(hStdOut, 7);

cout << "————————————————————————————————" << endl;

ch = \_getch(); //код нажатой клавиши

if (ch == -32) ch = \_getch(); // Отлавливаем стрелочки (2 байтовые)

switch (ch)

{

case ESCAPE:

return;

case UP:

if (active\_menu > 0) {

--active\_menu;

}

else {

active\_menu = size(Menu) - 1;

}

break;

case DOWN:

if (active\_menu < size(Menu) - 1) {

++active\_menu;

}

else {

active\_menu = 0;

}

break;

case SPACE:

switch (active\_menu)

{

case 4: rate\_Exam(choice, &clients[choice].englishGradeExam);

break;

case 3: rate\_Exam(choice, &clients[choice].mathGradeExam);

break;

case 0: rate\_Exam(choice, &clients[choice].oaipGradeExam);

break;

case 1: rate\_Exam(choice, &clients[choice].oopGradeExam);

break;

case 2: rate\_Exam(choice, &clients[choice].physicsGradeExam);

break;

case 5:

system("cls");

return;

}

break;

default:

break;

}

}

}

void sortNoutput() {

system("cls");

Users newArray[100];

for (int i = 0; i < CountOfUsers; ++i) {

newArray[i] = clients[i];

}

// shell

Users temp;

int gap;

for (gap = CountOfUsers / 2; gap > 0; gap /= 2) {

int i, j, sorted;

do

{

sorted = 0;

for (i = 0, j = gap; j < CountOfUsers; i++, j++)

if (newArray[i].averageGrade < newArray[j].averageGrade){

temp = newArray[i];

newArray[i] = newArray[j];

newArray[j] = temp;

sorted = 1;

}

} while (sorted);

}

int y = 6;

GoToXY(45, y);

cout << "| Топ";

GoToXY(52, y);

cout << "| Группа";

GoToXY(62, y);

cout << "| Ф.И.О.";

GoToXY(92, y);

cout << "| Логин";

GoToXY(117, y);

cout << "| Средняя отметка";

GoToXY(45, y + 1);

cout << "——————————————————————————————————————————————————————————————————————————————————————————";

y = 8;

for (int i = 0; i < 10 && newArray[i].group > 100000 && newArray[i].group < 999999 && newArray[i].averageGrade > 0; ++i, y += 2) {

GoToXY(45, y);

cout << "| " << i + 1;

GoToXY(52, y);

cout << "| " << newArray[i].group;

GoToXY(62, y);

cout << "| " << newArray[i].name;

GoToXY(92, y);

cout << "| " << newArray[i].login;

GoToXY(117, y);

cout.precision(3);

cout << "| " << newArray[i].averageGrade;

GoToXY(45, y + 1);

cout << "——————————————————————————————————————————————————————————————————————————————————————————";

}

GoToXY(70, y + 1);

system("pause");

system("cls");

}

void topStudents() {

averageGrade();

sortNoutput();

}

void averageGrade() {

for (int i = 0; i < CountOfUsers; ++i) {

clients[i].averageGrade = 0;

int marksCount = 0;

if (clients[i].englishGradeExam != 0) {

clients[i].averageGrade += clients[i].englishGradeExam;

++marksCount;

}

if (clients[i].englishGradeOlimp != 0) {

clients[i].averageGrade += clients[i].englishGradeOlimp;

++marksCount;

}

if (clients[i].mathGradeExam != 0) {

clients[i].averageGrade += clients[i].mathGradeExam;

++marksCount;

}

if (clients[i].mathGradeOlimp != 0) {

clients[i].averageGrade += clients[i].mathGradeOlimp;

++marksCount;

}

if (clients[i].oaipGradeExam != 0) {

clients[i].averageGrade += clients[i].oaipGradeExam;

++marksCount;

}

if (clients[i].oaipGradeOlimp != 0) {

clients[i].averageGrade += clients[i].oaipGradeOlimp;

++marksCount;

}

if (clients[i].oopGradeExam != 0) {

clients[i].averageGrade += clients[i].oopGradeExam;

++marksCount;

}

if (clients[i].oopGradeOlimp != 0) {

clients[i].averageGrade += clients[i].oopGradeOlimp;

++marksCount;

}

if (clients[i].physicsGradeExam != 0) {

clients[i].averageGrade += clients[i].physicsGradeExam;

++marksCount;

}

if (clients[i].physicsGradeOlimp != 0) {

clients[i].averageGrade += clients[i].physicsGradeOlimp;

++marksCount;

}

int j = 0;

while (clients[i].englishGradeLaba[j] != 0) {

clients[i].averageGrade += clients[i].englishGradeLaba[j];

++j;

++marksCount;

}

j = 0;

while (clients[i].mathGradeLaba[j] != 0) {

clients[i].averageGrade += clients[i].mathGradeLaba[j];

++j;

++marksCount;

}

j = 0;

while (clients[i].oaipGradeLaba[j] != 0) {

clients[i].averageGrade += clients[i].oaipGradeLaba[j];

++j;

++marksCount;

}

j = 0;

while (clients[i].oopGradeLaba[j] != 0) {

clients[i].averageGrade += clients[i].oopGradeLaba[j];

++j;

++marksCount;

}

j = 0;

while (clients[i].physicsGradeLaba[j] != 0) {

clients[i].averageGrade += clients[i].physicsGradeLaba[j];

++j;

++marksCount;

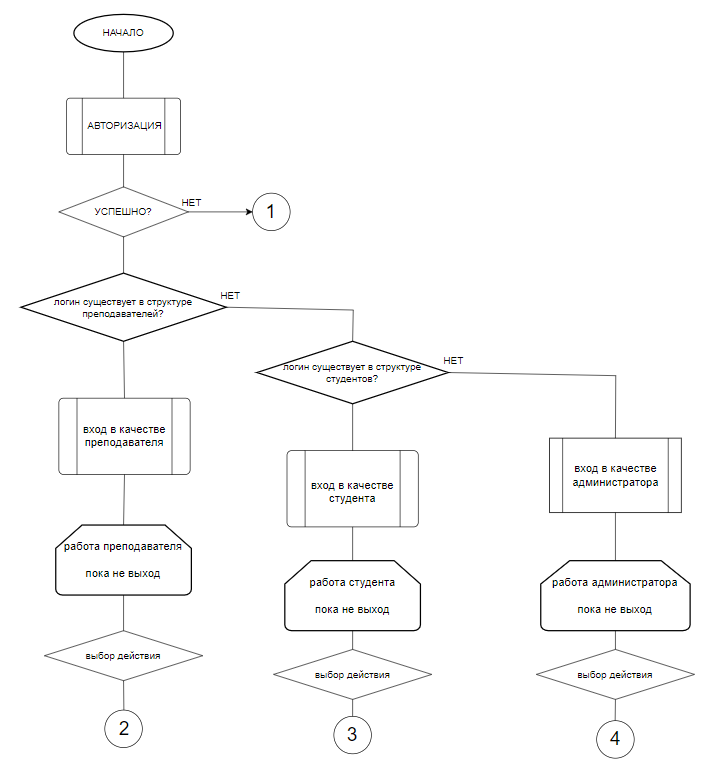
}

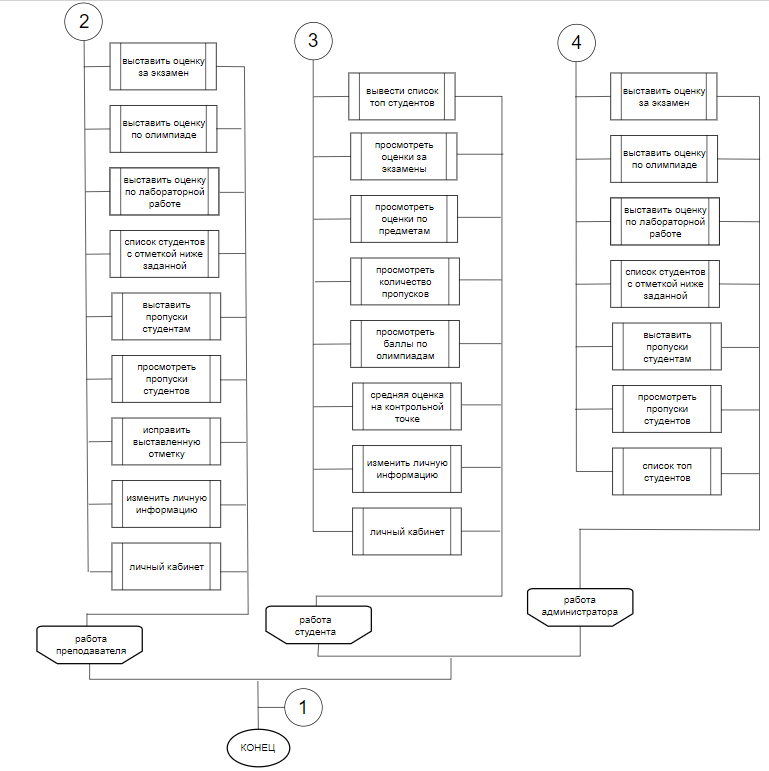
if (marksCount == 0) clients[i].averageGrade = 0;

else clients[i].averageGrade /= marksCount;

}

## ПРИЛОЖЕНИЕ В (справочное) Схема общего алгоритма работы программы





## ПРИЛОЖЕНИЕ Г (обязательное) Ведомость документов курсовой работы

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | | Наименование | | | | Дополнитель-ные сведения | |
|  | | | | | Текстовые документы | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | Пояснительная записка | | | | 46 с. | |
|  | | | | |  | | | |  | |
|  | | | | | Графические документы | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | Чертёж Контекстная диаграмм | | | | Формат А4 | |
|  | | | | | IDEF0 | | | |  | |
|  | | | | | Чертёж Алгоритм функции main | | | | Формат А4 | |
|  | | | | | Плакат Диаграмма BPMN | | | | Формат А4 | |
|  | | | | | «Учёт сведений о пациентах | | | |  | |
|  | | | | | медицинского центра» | | | |  | |
|  | | | | | Плакат Скриншоты рабочих окон | | | | Формат А4 | |
|  | | | | | программного средства | | | |  | |
|  | | | | | Плакат Модели представления | | | | Формат А4 | |
|  | | | | | программного средства | | | |  | |
|  | | | | | Плакаты, отражающие результаты | | | | Формат А4 | |
|  | | | | | проектирования программного | | | |  | |
|  | | | | | средства | | | |  | |
|  | | | | |  | | | |  | |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Разработка программы учета успеваемости студента |  | | | Лист | Листов |
| Разраб. | | Бадей П.Ю. |  |  |  |  |  | 1 | 1 |
| Пров. | | Карпейчик А.А. |  |  | Кафедра ЭИ  гр. 272303 | | | | |
|  | |  |  |  |
| Н.контр. | | Карпейчик А.А. |  |  |
|  | |  |  |  |