МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

#### Факультет информационных технологий и компьютерной безопасности

(факультет)

### Кафедра искусственного интеллекта и цифровых технологий

# КУРСОВОЙ ПРОЕКТ

по дисциплине Объектно-ориентированное программирование

тема Разработка программного обеспечения с использованием объектно-ориентированного подхода

**Расчетно-пояснительная записка**

Разработал студент А.О. Анохин

Подпись, дата Инициалы, фамилия

## Руководитель Б.Н. Тишуков

Подпись, дата Инициалы, фамилия

## Члены комиссии

Подпись, дата Инициалы, фамилия

Подпись, дата Инициалы, фамилия

Нормоконтролер Б.Н. Тишуков Подпись, дата Инициалы, фамилия

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                               дата

2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

### Кафедра искусственного интеллекта и цифровых технологий

ЗАДАНИЕ

на курсовой проект

по дисциплине Объектно-ориентированное программирование

тема Разработка программного обеспечения с использованием объектно-ориентированного подхода

Студент группы бТИИ-241 Анохин Антон Олегович

Фамилия, имя, отчество

Вариант 2. Складское хозяйство

Технические условия процессор AMD Ryzen 5 2600 3.40 ГГц, операционная система Windows 10, ОЗУ 24576 МБ

Содержание и объем проекта (графические работы, расчеты и прочее):

анализ предметной области и требований к программному обеспечению (16 страниц); моделирование и разработка системы на основе принципов ООП (6 страниц); реализация системы на общесистемном языке программирования (7 страниц); 7 рисунков, 0 таблиц, 1 приложение

Сроки выполнения этапов анализ предметной области и требований к программному обеспечению (01.03 – 25.03); моделирование и разработка системы на основе принципов ООП (26.03 – 20.04); реализация системы на общесистемном языке программирования (21.04 – 13.05); описание диалога с пользователем (14.05 – 21.05); оформление пояснительной записки (22.05 – 04.06)

### Срок защиты курсового проекта

Руководитель Б.Н. Тишуков

Подпись, дата Инициалы, фамилия

Задание принял студент А.О. Анохин

Подпись, дата Инициалы, фамилия

Замечания руководителя

# Содержание

|  |  |
| --- | --- |
| Введение………………………………………………………………….………….. | 5 |
| 1 Анализ предметной области……………………………………………………… | 6 |
| 1.1 Особенности предметной области…………..………………………………. | 7 |
| 1.2 Описание бизнес-процессов в рамках предметной области…..…………... | 11 |
| 1.3 Технологический процесс приёмки товаров с применением цифровых технологий………………………………………………………………………… | 16 |
| 1.4 Проблемы, возникающие в работе складского хозяйства и перспективы их решения с использованием программных средств......................................... | 19 |
| 2 Моделирование и разработка системы на основе принципов ООП ………….. | 22 |
| 2.1 Постановка задачи……………………………………………………………. | 23 |
| 2.2 Разработка классов ……………………………..……………………………. | 25 |
| 3 Реализация системы на общесистемном языке программирования…………… | 28 |
| 3.1 Выбор средств программной реализации…………………………………… | 28 |
| 3.2 Алгоритм выполнения программы………………………………………….. | 29 |
| 3.3 Модульная структура программы…………………………………………… | 31 |
| 3.4 Описание диалога с пользователем………………………………………… | 32 |
| Заключение…………………………………………………………………………... | 35 |
| Список использованной литературы………………………………………………. | 36 |
| Приложение ...……………………………………………………………………….. | 37 |

Введение

Тема курсового проекта: «Разработка программного обеспечения для управления складским хозяйством с использованием объектно-ориентированного подхода»

Цель работы: «Разработка программного комплекса для автоматизации учётных процессов на складе предприятия средствами объектно-ориентированного программирования»

В условиях современной рыночной экономики эффективное управление складскими запасами играет ключевую роль в обеспечении бесперебойной работы любого торгового или производственного предприятия. Складское хозяйство представляет собой сложную систему, требующую точного учёта товарно-материальных ценностей, контроля их движения и оптимального использования складских площадей. Однако на многих предприятиях до сих пор преобладают ручные методы учёта, которые характеризуются высокой трудоёмкостью, низкой оперативностью и повышенной вероятностью ошибок. Автоматизация складских процессов является актуальной задачей, решение которой позволяет существенно повысить эффективность управления материальными потоками. Разрабатываемый программный комплекс предназначен для автоматизации основных операций складского учёта: приёмки и отгрузки товаров, контроля остатков, управления складскими ячейками, проведения инвентаризации и формирования отчётной документации. Особое внимание уделяется созданию интуитивно понятного интерфейса и обеспечению надёжного хранения данных.

Практическая значимость работы заключается в том, что готовый программный продукт может быть внедрён на малых и средних предприятиях различных сфер деятельности, где требуется организация эффективного учёта товарно-материальных ценностей. Использование данной системы позволит оптимизировать складские операции и повысить общую эффективность работы предприятия.

# 1. Анализ предметной области

Логистика и складское хозяйство представляют собой одну из ключевых инфраструктурных отраслей современной экономики, выступая связующим звеном между производителями и потребителями товаров. Эффективность работы всего звена поставок напрямую зависит от слаженности и точности процессов, происходящих на складе — узловом пункте, где товары аккумулируются, хранятся, комплектуются и распределяются далее по цепочке. В условиях растущих объемов товарооборота, увеличения номенклатуры и повышения требований к скорости обслуживания склад перестает быть просто местом для хранения, превращаясь в сложный технологический и управленческий комплекс. Его работа требует координации материальных, информационных и человеческих потоков, что делает эту предметную область чрезвычайно многогранной и насыщенной данными. Анализ этой области позволяет выявить системные проблемы, узкие места и потенциальные точки роста, для устранения которых все активнее применяются специализированные программные средства, в том числе построенные на принципах объектно-ориентированного проектирования, позволяющего адекватно отразить структуру и взаимосвязи реальных складских сущностей.

# 1.1 Особенности предметной области

Складское хозяйство представляет собой сложную организационно-техническую систему, обеспечивающую процессы хранения, учёта и распределения материальных ценностей. Анализ предметной области показывает, что современный склад – это не просто помещение для хранения товаров, а важнейшее звено в логистической цепи предприятия, от эффективности работы которого напрямую зависит успех всей хозяйственной деятельности.

Основная функция складского хозяйства заключается в создании условий для сохранности товарно-материальных ценностей и обеспечении бесперебойного снабжения производственных процессов или торговых операций. Однако в современных экономических условиях круг задач, решаемых складским комплексом, значительно расширился. Сегодня склад выполняет не только функции хранения, но и активно участвует в процессах сортировки, комплектации, маркировки, упаковки и подготовки товаров к отгрузке.

Ключевой особенностью складской деятельности является её двойственная природа. С одной стороны, склад представляет собой центр затрат, требующий значительных финансовых вложений в содержание помещений, оборудования и персонала. С другой стороны, правильно организованное складское хозяйство способно генерировать дополнительную прибыль за счёт оптимизации товарных запасов, сокращения сроков хранения и минимизации потерь.

Технологический процесс на складе характеризуется чёткой цикличностью и включает в себя несколько взаимосвязанных этапов. Начинается он с приёмки товаров, которая содержит проверку количества и качества поступающей продукции, оформление сопроводительной документации и размещение на временное хранение. Далее следует этап собственно хранения, в ходе которого обеспечиваются оптимальные условия содержания товаров с учётом их физико-химических свойств и сроков годности. Завершающим этапом является отгрузка, включающая подборку заказов, комплектацию партий и оформление отгрузочных документов.

Учётная система на складе требует особого внимания к точности и оперативности отражения хозяйственных операций. Каждое движение товаров должно фиксироваться в реальном времени, что позволяет поддерживать актуальность данных об остатках и обеспечивать достоверность складской отчётности. Особую сложность представляет учёт товаров с ограниченным сроком годности, требующий внедрения системы ротации запасов по принципу FIFO (First In, First Out) или FEFO (First Expired, First Out).

Организационная структура складского хозяйства обычно включает несколько функциональных зон: зону приёмки, зону хранения, зону комплектации и зону отгрузки. Каждая из этих зон имеет свои особенности оснащения и организации рабочих процессов. Рациональное зонирование складского пространства позволяет оптимизировать маршруты перемещения товаров и сократить временные затраты на выполнение операций.

Информационные потоки на складе отличаются высокой интенсивностью и разнонаправленностью. Информация о поступлении товаров, их перемещении между ячейками хранения, отгрузке покупателям должна своевременно поступать в систему учёта и находить отражение в соответствующих регистрах. При этом важно обеспечить согласованность данных складского учёта с данными бухгалтерского и управленческого учёта.

Персонал склада играет ключевую роль в обеспечении эффективности складских процессов. Работники склада должны обладать не только физической выносливостью, но и определёнными профессиональными знаниями и навыками, включая умение работать со складской техникой, знание системы учёта и особенностей хранимой продукции. Мотивация персонала и организация труда напрямую влияют на производительность складских операций.

Внешние факторы, такие как сезонные колебания спроса, изменения рыночной конъюнктуры, требования законодательства в области хранения определенные категории товаров, также оказывают значительное влияние на организацию складского хозяйства. Гибкость и адаптивность становятся ключевыми качествами современного склада, способного быстро реагировать на изменение внешних условий.

Техническое оснащение современного склада представляет собой комплекс взаимосвязанных элементов, включающий складское оборудование, подъёмно-транспортные механизмы, средства автоматизации и вычислительную технику. От правильного выбора и рациональной расстановки технических средств зависит эффективность использования складских площадей и производительность труда персонала. Особое значение имеет внедрение систем автоматической идентификации товаров (штрих-кодирование, RFID-метки), которые позволяют значительно ускорить процессы приёмки, отгрузки и инвентаризации.

Экономические аспекты складской деятельности требуют тщательного расчёта затрат на содержание складских помещений, амортизацию оборудования, оплату труда персонала и другие операционные расходы. Важнейшей задачей является оптимизация размера товарных запасов, поскольку как их избыток, так и недостаток приводят к негативным экономическим последствиям. Избыточные запасы связывают оборотные средства предприятия и увеличивают затраты на хранение, тогда как их недостаток может привести к срыву производственных программ или потерям продаж.

Безопасность складской деятельности включает несколько аспектов: сохранность товарно-материальных ценностей, пожарную безопасность, охрану труда персонала и информационную безопасность учётных данных. Каждый из этих аспектов требует разработки и внедрения соответствующих регламентов и технических средств контроля. Особенно актуальной в последнее время становится задача защиты конфиденциальной информации о товарных остатках и движении ценностей.

Экологические требования к организации складского хозяйства становятся всё более строгими, особенно при хранении опасных грузов, химической продукции или товаров, требующих специальных условий утилизации. Соблюдение экологических стандартов предполагает не только выполнение законодательных норм, но и внедрение ресурсосберегающих технологий, снижающих воздействие на окружающую среду.

Взаимодействие с другими подразделениями предприятия является важным условием эффективной работы склада. Складская служба должна тесно сотрудничать с отделом закупок по вопросам своевременного пополнения запасов, с отделом продаж – по вопросам отгрузки готовой продукции, с производственными подразделениями – по вопросам снабжения материалами. Координация этих взаимодействий требует хорошо зарекомендовавшей себя системы документооборота и чёткого распределения ответственности.

Перспективы развития складского хозяйства связаны с внедрением технологий Industry 4.0, включая интернет вещей (IoT), аналитику больших данных, роботизацию складских процессов и искусственный интеллект для прогнозирования спроса. Цифровая трансформация складской деятельности открывает новые возможности для повышения эффективности, но одновременно требует значительных инвестиций в модернизацию и подготовку персонала.

Таким образом, современное складское хозяйство представляет собой сложную многоаспектную систему, эффективное управление которой требует комплексного подхода и использования передовых информационных технологий. Автоматизация учётных процессов является не просто желательным улучшением, а необходимым условием для повышения конкурентоспособности предприятия в современных экономических условиях.

# 1.2 Бизнес-процессы складского хозяйства и их анализ

Основные бизнес-процессы складского хозяйства можно структурировать по их функциональной направленности и последовательности выполнения. Каждый процесс представляет собой совокупность взаимосвязанных операций, направленных на достижение конкретной цели складской деятельности.

Процесс приёмки товаров является начальным этапом складского цикла и включает следующие ключевые операции:

1. Проверка сопроводительных документов (накладных, счетов-фактур, сертификатов качества)
2. Сверка фактического количества товара с указанным в документах
3. Контроль качества поступающей продукции
4. Временное размещение в зоне приёмки
5. Внесение данных в систему учёта

Особенностью данного процесса является необходимость оперативного принятия решений в случае выявления расхождений по количеству или качеству товаров. Задержки на этапе приёмки могут привести к срыву последующих операций и нарушению ритмичности работы склада.

Процесс размещения на хранение включает в себя определение оптимального места для хранения товара с учётом его характеристик:

1. Выбор зоны хранения (холодильная, отапливаемая, секция для опасных грузов)
2. Определение конкретной ячейки с учётом габаритов товара
3. Обеспечение условий хранения (температурный режим, влажность, освещённость)
4. Маркировка места хранения
5. Обновление данных в системе управления складом

Критически важным аспектом данного процесса является максимизация использования складских площадей при обеспечении сохранности товаров и удобства их дальнейшего изъятия.

Процесс хранения и обеспечения сохранности включает:

1. Регулярный контроль условий хранения
2. Мониторинг сроков годности
3. Проведение профилактических мероприятий
4. Обеспечение безопасности (противопожарные меры, охрана)
5. Систематизацию товаров по принципу ротации

Особую сложность представляет управление товарами с ограниченным сроком годности, требующее внедрения сложной системы мониторинга и своевременного списания.

Процесс отбора и комплектации заказов является одним из наиболее трудоёмких:

1. Получение заявок от отдела продаж или производственных подразделений
2. Планирование маршрутов отбора товаров
3. Физический отбор товаров с мест хранения
4. Проверка соответствия отобранного товара заказу
5. Упаковка и маркировка для отгрузки

Эффективность данного процесса напрямую влияет на удовлетворённость клиентов и требует оптимизации по времени выполнения и точности комплектации.

1. Процесс отгрузки товаров завершает складской цикл:
2. Оформление сопроводительных документов
3. Погрузка на транспортные средства
4. Сверка отгружаемого товара с документами
5. Архивация документов по отгрузке
6. Корректировка остатков в системе учёта

На данном этапе особенно важна чёткая координация с транспортной службой и соблюдение графика отгрузок.

Вспомогательные процессы включают инвентаризацию, техническое обслуживание оборудования, обучение персонала и другие обеспечивающие функции. Регулярное проведение инвентаризации позволяет выявить расхождения между учётными и фактическими данными и является важным инструментом контроля.

Анализ узких мест в складских процессах показывает, что наиболее проблемными зонами являются:

1. Ручной ввод данных и бумажный документооборот
2. Отсутствие информации о состоянии запасов в реальном времени
3. Неоптимальная организация складского пространства
4. Недостаточная координация между смежными подразделениями

Решение этих проблем требует внедрения интегрированной системы управления складом, способной автоматизировать ключевые бизнес-процессы и обеспечить прозрачность информации на всех этапах складского цикла.

Подготовительный этап начинается с получения уведомления о поставке. Кладовщик проверяет соответствие графика поставок, выделяет необходимые ресурсы — персонал, погрузочную технику, зону для временного размещения. Особое внимание уделяется подготовке документации: формируются товарные ярлыки, проверяются бланки актов расхождений.

Разгрузка и первоначальный осмотр проводятся с обязательным участием представителя поставщика. Каждая единица товара сверяется с сопроводительной накладной по наименованию и количеству. Выборочно проверяется целостность упаковки, соответствие маркировки. При обнаружении нарушений составляется акт с участием всех сторон.

Количественная приёмка требует скрупулёзного подсчета. Для штучных товаров применяется поштучный пересчет, для насыпных — взвешивание с оформлением весовой накладной. Сложности возникают при приемке нефасуемой продукции, где допустимы естественные нормы расхождения.

Качественная экспертиза включает проверку сертификатов соответствия, сроков годности, органолептических показателей. Для технически сложных товаров может потребоваться выборочная сборка и тестирование. Результаты фиксируются в журнале входного контроля.

Окончательное оформление заключается в подписании приемочных документов, простановке штампов в сопроводительных документах. Информация вносится в журналы складского учета, товар присваивается инвентарные номера.

**Процесс размещения на хранение: системный подход**

Зонирование склада основывается на классификации товаров по габаритам, весу, оборачиваемости. Тяжелые и крупногабаритные грузы размещаются ближе к зонам отгрузки, товары частого спроса — в легкодоступных местах. Сезонная продукция распределяется с учетом прогноза спроса.

Оптимизация пространства требует расчета коэффициента использования объема. Стеллажные системы подбираются под специфику товаров — паллетные стеллажи для групповых упаковок, консольные для длинномерных грузов, мобильные для архивного хранения.

Маркировка и идентификация включают присвоение адресного кода ячейке, нанесение визуальных меток, внесение данных в систему навигации. Для опасных грузов дополнительно обозначаются зоны специальными знаками безопасности.

**Процесс хранения: ежедневные операции**

Контроль условий хранения предполагает регулярные обходы территории, проверку работы климатического оборудования, ведение журналов температуры и влажности. Для скоропортящихся товаров устанавливается график внеплановых проверок.

Ротация запасов осуществляется по утвержденным графикам. Система FIFO требует строгого соблюдения очередности отгрузки, что усложняется при большом ассортименте. Персонал проходит специальное обучение методам визуального контроля сроков годности.

Обеспечение сохранности включает противопожарные мероприятия, организацию пропускного режима, установку систем видеонаблюдения. Особое внимание уделяется правилам штабелирования для предотвращения аварийных ситуаций.

**Процесс отбора и комплектации: технологические нюансы**

Обработка заявок начинается с проверки остатков, резервирования товара в системе. Формируются маршрутные листы с оптимальной последовательностью отбора. Сложные заказы с большим ассортиментом распределяются между несколькими комплектовщиками.

Физический отбор требует точного соблюдения количественных показателей. Используются средства малой механизации — штабелеры, тележки с подъемными устройствами. Для хрупких товаров разрабатываются специальные инструкции по обращению.

Контроль качества комплектации включает выборочную проверку случайных заказов, сверку с оригинальной заявкой. Обнаруженные ошибки фиксируются в системе для анализа причин возникновения.

Процесс отгрузки: финальная стадия

Подготовка документов включает оформление товарно-транспортных накладных, счетов-фактур, сертификатов. Проводится сверка юридических реквизитов, условий поставки.

Погрузочные операции требуют координации действий водителя и грузчиков. Осуществляется контроль правильности размещения груза в транспортном средстве, крепления для длительных перевозок.

Информационное сопровождение заключается во внесении отметок об отгрузке в систему учета, отправке уведомлений получателю, архивации документов. Формируются отчеты о выполненных отгрузках за смену.

Каждый процесс регламентируется внутренними инструкциями, которые регулярно актуализируются с учетом изменений в технологии работы. Персонал проходит ежегодное обучение с практической отработкой действий в нестандартных ситуациях.

# 1.3 Технологический процесс приёмки товаров с применением цифровых технологий

Современный процесс приёмки товаров на складе, оснащённом цифровыми системами, представляет собой высокоорганизованный и автоматизированный конвейер, направленный на минимизацию ошибок, ускорение обработки и обеспечение полной прослеживаемости каждой товарной единицы. Внедрение таких технологий трансформирует традиционные, часто бумажные, процедуры в стройный цифровой рабочий процесс, где каждый этап контролируется программным обеспечением.

Процесс начинается с предварительного уведомления от поставщителя. Заблаговременно, ещё до физического прибытия груза, на склад поступает электронная накладная. Этот документ, отправленный через защищённые каналы связи, содержит исчерпывающий перечень ожидаемого товара: его уникальные идентификаторы, наименования, заявленное количество, данные о сериях и сроках годности. Система управления складом автоматически регистрирует это уведомление, создавая в своей базе данных «ожидаемые» позиции. Это формирует цифровой образ предстоящей поставки, с которым будет сверяться реальный груз. Данный этап иллюстрирует принцип абстракции, присущий объектно-ориентированному подходу: система оперирует не физическими коробками, а их формализованными цифровыми представлениями — сущностями, обладающими конкретными атрибутами.

Непосредственная приёмка происходит в зоне разгрузки. Сотрудник, вооружённый мобильным терминалом или планшетом со специализированным программным обеспечением, сканирует штрихкод или RFID-метку с каждой прибывшей единицы товара. В этот момент цифровая система оживания получает физическое подтверждение. Сканирование — это не просто считывание числа; это запрос к системе на поиск соответствующей цифровой записи. Система мгновенно сверяет полученный идентификатор с данными из электронной накладной. При совпадении запись о товаре переходит из статуса «ожидается» в статус «принят». Если система обнаруживает несоответствие — например, не заказанный товар или расхождение в количестве — она немедленно сигнализирует об этом оператору визуальным и звуковым оповещением, приостанавливая процесс для ручного разрешения ситуации. Здесь проявляется принцип инкапсуляции: логика проверки и изменения статуса инкапсулирована внутри системы, пользователю предоставляется лишь интерфейс для взаимодействия и результат операции.

Следующий критически важный этап — проверка качества и соответствия. Помимо сверки номенклатуры, система может инициировать контроль сроков годности. Оператор может выборочно или сплошняком проверить маркировку на упаковках, а результаты вносить прямо в терминал. Просроченный или не соответствующий стандартам качества товар немедленно маркируется в системе как «брак» или «на доработку/утилизацию», и для него автоматически блокируется возможность перемещения в основную зону хранения. Таким образом, цифровой барьер не позволяет некондиционному товару проникнуть в товарооборот.

После успешного прохождения контроля система определяет оптимальное место для размещения товара на складе. Это сложная логистическая задача, которая решается автоматически на основе заложенных алгоритмов. Система анализирует множество параметров: категорию товара (для размещения в тематических зонах), его габариты и вес (для выбора стеллажа соответствующей грузоподъёмности), срок годности (для реализации принципа «первым пришёл — первым ушёл» или FIFO), а также текущую заполненность и топологию складских ячеек. Алгоритм стремится максимизировать использование пространства, минимизировать время на последующий отбор и обеспечить сохранность груза. Выбрав подходящую ячейку, система направляет оператора точно к ней, например, через световую индикацию на стеллажах или маршрут на карте терминала. При этом цифровой статус ячейки меняется с «свободна» на «занята», а между записью о товаре и записью о ячейке устанавливается неразрывная цифровая связь. Это демонстрирует принцип установления отношений между сущностями, фундаментальный для объектно-ориентированного моделирования: товар и ячейка становятся связанными объектами в системе.

Финальным аккордом процесса является автоматическое обновление учётных записей и формирование отчётности. Количество товара на складе увеличивается в реальном времени, что сразу же отражается в системе учёта и становится видимым для всех отделов, от закупок до продаж. Всё перемещение товара — от двери зоны получения до конкретной полки — фиксируется в системе как законченная транзакция с временными метками и ответственными лицами. Это обеспечивает абсолютную прозрачность и создаёт неизменяемый аудитный след для любого последующего анализа.

Таким образом, цифровизация процесса приёмки товаров позволяет перейти от реактивного ручного труда к проактивному управлению, управляемому данными. Каждая единица товара обретает свой цифровой двойник, жизненным циклом которого управляет система, обеспечивая высочайший уровень точности, скорости и контроля на современном складе.

# 1.4 Проблемы, возникающие в работе складского хозяйства и перспективы их решения с использованием программных средств

Несмотря на кажущуюся простоту базовых операций (принять, положить, взять, отгрузить), складское хозяйство сталкивается с целым комплексом проблем, большая часть которых порождена человеческим фактором, ограниченностью ресурсов и несовершенством традиционных методов управления.

Одной из наиболее острых и затратных проблем является низкая точность учета товаров. При ручном ведении журналов или использовании упрощенных табличных программ неизбежны описки, арифметические ошибки, забывчивость сотрудников. Расхождения между фактическим наличием товара на полках и данными в учетной системе (слепая зона склада) приводят к прямым финансовым убыткам: невозможно продать то, что якобы есть в системе, но отсутствует физически, или, наоборот, возникают излишки, ведущие к заморозке оборотных средств. Решение этой проблемы лежит в области автоматизированной идентификации и сбора данных. Программные комплексы, интегрированные со сканирующим оборудованием (штрихкоды, RFID), позволяют практически полностью исключить ручной ввод. Каждая операция — приемка, перемещение, отгрузка — подтверждается сканированием, что делает ее фиксацию в системе мгновенной и безошибочной. Цифровой след товара становится его единственным источником истины, а состояние системы всегда соответствует физическому состоянию склада.

Следующая группа проблем связана с неэффективным использованием складского пространства и ресурсов. Без помощи оптимизирующих алгоритмов размещение товаров часто происходит хаотично или по устаревшим шаблонам. Это приводит к тому, что быстрооборачиваемые товары оказываются в глубине склада, а медленные — у проходов, сотрудники тратят до 70% рабочего времени на бесцельные переходы, а полезный объем стеллажей используется нерационально. Современные системы управления складом (WMS) решают эту проблему с помощью сложных алгоритмов размещения. Эти алгоритмы, основанные на анализе данных об оборачиваемости, весо-габаритных характеристиках, сроках годности и логистике пополнения, рассчитывают оптимальное место для каждого товара. Они могут, например, группировать товары из одного заказа рядом друг с другом (зонирование), чтобы ускорить последующий отбор, или размещать тяжелые паллеты в нижних ярусах для безопасности. Таким образом, программное обеспечение трансформирует склад из хаотичного хранилища в высокоорганизованное пространство, где каждое решение по размещению принимается на основе данных, минимизируя издержки и максимизируя пропускную способность.

Еще одна критическая проблема — отсутствие оперативной аналитики и прогнозирования. В традиционной системе управленческие решения часто принимаются постфактум, на основе вчерашних отчетов или интуиции. Руководство может не знать о внезапном всплеске спроса на определенную группу товаров, что приводит к их дефициту, или, наоборот, пропустить момент накопления неликвидов. Программные средства, построенные на принципах объектно-ориентированного проектирования, где такие сущности, как «Товар», «Заказ», «Поставка», являются самостоятельными объектами с историей своих состояний, позволяют накапливать большие массивы структурированных данных. Дальнейший анализ этих данных с помощью встроенных инструментов отчетности и визуализации (дашборды, графики, сводки) предоставляет менеджерам мощный инструмент для принятия обоснованных решений. Система может автоматически формировать прогнозы спроса, рассчитывать оптимальный размер заказа, сигнализировать о необходимости пополнения запасов по критическим позициям (формирование заявок на закупку), тем самым переводя управление складом из реактивного режима в проактивный.

Наконец, высокая зависимость от квалификации персонала и сложность обучения также является серьезным вызовом. Уход ключевого сотрудника, долго работавшего по «уникальной» системе записей, может парализовать работу склада на несколько дней. Цифровые системы нивелируют эту проблему, формализуя и алгоритмизируя все бизнес-процессы. Новый сотрудник с минимальной подготовкой, вооруженный терминалом, который пошагово ведет его по процессу (сканируй это, затем иди туда, положи это), способен быстро влиться в работу и показывать высокую результативность с минимальным количеством ошибок. Программа выступает в роли безотказного цифрового наставника и контролера, стандартизируя операции и снижая порог входа для персонала.

Таким образом, проблемы традиционного складского учета носят системный характер и напрямую влияют на финансовые результаты компании. Их решение заключается не в простой механизации, а в глубокой цифровой трансформации, основой которой являются специализированные программные комплексы. Эти системы, особенно те, что построены на объектно-ориентированных принципах, позволяют адекватно смоделировать предметную область, связав цифровые объекты с их физическими прототипами, и за счет автоматизации, оптимизации и предоставления аналитики выводят управление складом на качественно новый уровень эффективности и предсказуемости.

# 2. Моделирование и разработка системы на основе принципов ООП

Переход от анализа предметной области к непосредственному проектированию программного обеспечения является ключевым этапом. Его цель — создать непротиворечивую, логичную и технологически реализуемую модель будущей системы, которая будет адекватно отражать все сущности и процессы, выявленные в ходе анализа. Объектно-ориентированный подход (ООП) идеально подходит для решения этой задачи, так как позволяет представить сложную предметную область в виде набора взаимодействующих объектов, инкапсулирующих данные и поведение. Данный раздел посвящен формализации требований к системе, проектированию её структуры и описанию применяемых объектно-ориентированных принципов.

# 2.1 Постановка задачи

На основе проведенного анализа предметной области была сформулирована задача на разработку программного модуля «Цифровая приёмка товаров», являющегося ключевым компонентом более крупной системы управления складом (WMS).

Цель разработки: Повышение эффективности, точности и скорости процесса приёмки товаров на складе за счёт автоматизации ключевых операций, минимизации человеческого фактора и обеспечения полной цифровой прослеживаемости каждой товарной единицы.

**Основные требования к функционалу модуля**:

1. *Автоматизированная регистрация поставок*: Система должна предоставлять интерфейс для предварительной регистрации ожидаемых поставок на основе электронных накладных (вручную или через интеграцию с системой поставщика).
2. *Пооперационный учёт при сканировании*: Каждая единица товара должна быть принята в систему исключительно путём сканирования её штрихкода. Ручной ввод должен быть сведён к абсолютному минимуму и использоваться только в исключительных ситуациях (например, повреждённый штрихкод).
3. *Контроль и валидация данных*: Система обязана в реальном времени сверять сканируемые данные с информацией из зарегистрированной накладной. При обнаружении несоответствий (незаказанный товар, расхождение в количестве) система должна блокировать дальнейшую приёмку данной позиции и уведомлять оператора.
4. *Управление браком и некондицией*: Должен быть реализован механизм оперативного перевода товара в статус «брак» или «на утилизацию» с обязательным указанием причины. Такой товар не должен быть размещён на основном хранении.
5. *Интеллектуальное размещение*: Система должна автоматически определять и рекомендовать оптимальную складскую ячейку для размещения принятого товара на основе заданных алгоритмов (учёт габаритов, оборачиваемости, срока годности, заполненности зон).
6. *Автоматическое обновление учётных данных*: Факт приёмки и размещения товара должен мгновенно отражаться в общем балансе склада, делая товар доступным для заказа и отгрузки.
7. *Формирование документации*: Система должна автоматически формировать и сохранять электронные акты приёмки и другие сопутствующие документы с указанием даты, времени и ответственного сотрудника.

**Нефункциональные требования**:

1. *Производительность*: Обработка данных и отклик интерфейса должны быть достаточно быстрыми для работы в режиме реального времени без задержек, мешающих оператору.
2. *Удобство использования*: Пользовательский интерфейс должен быть интуитивно понятным, минималистичным и адаптированным для работы на мобильном устройстве (планшете) в условиях склада.
3. *Надёжность*: Система должна обеспечивать целостность данных и предотвращать их потерю в случае сбоев оборудования или программного обеспечения.

# 2.2 Разработка классов

Для реализации поставленных задач была спроектирована объектная модель системы. Её ядро составляют следующие классы, непосредственно отражающие ключевые сущности предметной области:

**Класс: Product (Товар)**

Назначение: Основная бизнес-сущность. Описывает товарную позицию на складе.

Ключевые атрибуты:

1. id: Уникальный идентификатор (артикул) товара.
2. name: Наименование товара.
3. category: Категория, к которой относится товар.
4. amount: Количество единиц товара в наличии.
5. price: Цена за единицу товара.
6. expirationDays: Срок годности в днях (используется для логики проверки).
7. supplier: Наименование поставщика.
8. receiptDate: Дата поступления на склад (объект вложенной структуры SimpleDate).

Основные методы:

1. isExpired(): Проверяет, просрочен ли товар, на основе даты поступления и текущей даты.
2. printInfo(): Выводит отформатированную информацию о товаре в консоль.

**Класс: StorageCell (Ячейка хранения)**

Назначение: Описывает физическую ячейку стеллажа, где хранятся товары. Отвечает за учёт занятого места.

Ключевые атрибуты:

1. id: Уникальный идентификатор ячейки.
2. location: Строковое описание местоположения (например, "Стеллаж A-1").
3. maxCapacity: Максимальная вместимость ячейки.
4. currentProduct: Указатель на товар (const Product\*), хранящийся в ячейке.
5. currentAmount: Текущее количество единиц товара в ячейке.

Основные методы:

1. reserve(const Product\* product, int amount): Основной метод для размещения (резервирования места под) товара в ячейке. Выполняет проверки на корректность операции.
2. isEmpty(): Проверяет, свободна ли ячейка.

**Класс: Warehouse (Склад)**

Назначение: Центральный класс-контроллер. Управляет коллекциями товаров (products) и ячеек (cells), а также логикой их взаимодействия.

Ключевые атрибуты:

1. name: Название склада.
2. address: Физический адрес склада.
3. products: Вектор, содержащий все товары на складе.
4. cells: Вектор, содержащий все ячейки склада.

Основные методы:

1. addProduct(), addCell(): Добавляют новые товары и ячейки в соответствующие векторы.
2. findProductById(), findProductByName(): Поиск товаров по ID и имени (с поддержкой частичного совпадения).
3. placeProduct(): Связывает товар и ячейку, вызывая метод reserve() у ячейки и обновляя общее количество товара.
4. conductInventory(): Запускает процесс инвентаризации — выводит информацию по всем ячейкам.

**Класс: User (Пользователь)**

Назначение: Моделирует учётную запись пользователя системы для разграничения прав доступа.

Ключевые атрибуты:

1. login: Логин пользователя.
2. passwordHash: Пароль (в данной реализации хранится открыто, что является уязвимостью).
3. role: Роль пользователя (тип UserRole), определяющая его права в системе.

Основные методы:

1. authenticate(): Проверяет совпадение введённых логина и пароля с данными пользователя.

**Класс: FileDatabase (Работа с файлами)**

Назначение: Отвечает за сохранение состояния данных программы в файлы и их последующую загрузку.

Ключевые атрибуты:

1. basePath: Путь к директории, где хранятся файлы данных.

Основные методы:

1. saveProducts(): Сохраняет все товары в текстовый файл products.txt.
2. saveProductCellRelations(): Сохраняет информацию о том, какой товар и в каком количестве хранится в каждой ячейке, в файл product\_cells.txt.
3. loadProducts(): Заглушка. Предназначена для загрузки товаров из файла (требует реализации).

**Структура: SimpleDate (Упрощённая дата)**

Назначение: Вспомогательная структура для работы с датой поступления товара.

Поля: day, month, year.

Методы:

1. toString(): Преобразует дату в строку для удобного вывода и сохранения.
2. isExpired(): Упрощённая логика. Проверяет, истёк ли год годности.

**Перечисление: UserRole (Роль пользователя)**

Назначение: Определяет возможные роли пользователей в системе для управления доступом (Storekeeper - Кладовщик, Manager - Менеджер, Administrator - Администратор).

# 3. Реализация системы на общесистемном языке программирования

Разработанная система управления складом (СКД) реализована на языке C++. Этот выбор обусловлен его высокой производительностью, возможностью низкоуровневого управления памятью, что важно для ресурсоёмких операций с большими объёмами данных на складе, а также богатой стандартной библиотекой, предоставляющей необходимые структуры данных и средства ввода-вывода.

# 3.1 Выбор средств программной реализации

1. Для разработки проекта использовался следующий набор инструментов и технологий:
2. Язык программирования: C++ (стандарт C++11 или выше). Это компилируемый язык, что обеспечивает высокую скорость выполнения программы, критически важную для оперативного управления складскими процессами.
3. Среда разработки и компилятор: Проект может быть собран с помощью любого современного компилятора, поддерживающего стандарт C++ (например, GCC, Clang или MSVC). Разработка велась в среде Visual Studio 2022 для Windows, но код является кроссплатформенным и может быть скомпилирован под Linux или macOS.
4. Ключевые модули стандартной библиотеки C++:
   1. <iostream>: Для реализации консольного ввода и вывода информации.
   2. <vector>: Для хранения динамических массивов товаров (vector<Product>) и ячеек хранения (vector<StorageCell>). Это основная структура данных для управления списками объектов.
   3. <string>: Для работы со строковыми данными (наименования, категории, поставщики).
   4. <fstream>: Для реализации функций сохранения данных о товарах и ячейках в текстовые файлы.
   5. <stdexcept>: Для использования стандартных исключений (например, runtime\_error) для обработки ошибок, что делает код более надежным.
5. Парадигмы программирования: Используется объектно-ориентированное программирование (ООП). Это позволяет инкапсулировать логику работы с товарами, ячейками и складом в отдельные классы, что повышает читаемость, сопровождаемость и масштабируемость кода.

# 3.2 Алгоритм выполнения программы

Общая последовательность работы программы выглядит следующим образом:

1. Инициализация: Программа запускается, выполняется функция main().
   1. Устанавливается русская кодировка консоли (system("chcp 1251")).
   2. Создается объект базы данных FileDatabase и основной объект Warehouse (склад) с названием и адресом.
   3. В память программы загружается список пользователей системы с их ролями.
   4. Для демонстрации работы в векторы склада добавляются тестовые данные: несколько товаров (молоко, хлеб, сыр) и несколько ячеек хранения.
2. Аутентификация пользователя:
   1. Пользователю предлагается ввести свой логин и пароль.
   2. Происходит проверка введенных данных против списка загруженных пользователей.
   3. Если данные верны, определяется роль пользователя и начинается основная работа. Если нет — программа завершается с ошибкой.
3. Работа с главным меню:
   1. После успешного входа на экран выводится главное текстовое меню с пунктами.
   2. Пользователь выбирает номер нужного ему действия.
   3. В зависимости от выбора управление передается соответствующему блоку кода (case внутри switch).
   4. После выполнения выбранного действия программа снова возвращается к показу меню. Цикл продолжается до тех пор, пока пользователь не выберет пункт "Выход".
4. Завершение работы:
   1. При выходе пользователь может выбрать сохранение измененных данных в файлы.
   2. Программа корректно завершает свою работу.

Блок-схема алгоритма работы главного меню:

# 

# Рисунок 1 — Блок-схема алгоритма программы

# 3.3 Модульная структура программы

Программа имеет модульную структуру, где каждый класс и структура вынесены в отдельный логический блок. Это не заголовочные файлы (.h), а описание логических модулей внутри одного файла для удобства студенческой разработки.

1. **Модуль данных (Data Structures)**:
   1. SimpleDate: Структура для работы с датами.
   2. UserRole: Перечисление для определения ролей.
   3. Назначение: Эти сущности являются фундаментом для основных классов, определяя типы данных и состояния.
2. **Модуль бизнес-логики (Business Logic)**:
   1. Product: Отвечает за данные и поведение товара.
   2. StorageCell: Отвечает за логику размещения и хранения товаров в ячейках.
   3. Warehouse: Центральный модуль, который управляет всеми операциями: добавление, поиск, инвентаризация. Координирует работу между товарами и ячейками.
   4. Назначение: Это ядро программы, реализующее все основные функции системы складского учета.
3. **Модуль авторизации (Authorization):**
   1. User: Отвечает за хранение учетных данных и аутентификацию.
   2. Назначение: Обеспечивает безопасность и разграничение прав доступа к системе.
4. **Модуль работы с данными (Data Persistence):**
   1. FileDatabase: Отвечает за сохранение состояния программы в файлы и его восстановление.
   2. Назначение: Обеспечивает долговременное хранение данных между сеансами работы программы.
5. **Модуль пользовательского интерфейса (User Interface):**
   1. Функции printMenu(), pressEnterToContinue().
   2. Блок switch в функции main(), обрабатывающий выбор пользователя.
   3. Назначение: Обеспечивает взаимодействие между пользователем и системой через консоль.

Схема взаимодействия модулей выглядит следующим образом: пользователь через UI-модуль дает команды модулю бизнес-логики (Warehouse), который, в свою очередь, manipлирует объектами Product и StorageCell. При необходимости сохранения данных Warehouse обращается к модулю FileDatabase.

# 3.4 Описание диалога с пользователем

Взаимодействие с системой происходит через консольный интерфейс (CLI). Диалог построен по принципу последовательного вывода меню и ожидания ввода от пользователя.

**1. Экран аутентификации:**

Вывод программы: "=== СИСТЕМА УПРАВЛЕНИЯ СКЛАДОМ === Логин: Пароль:"

Действие пользователя: Вводит логин и пароль. Например, admin и root.

Реакция системы: При успешной проверке выводит приветствие "Добро пожаловать, admin!". При ошибке — сообщение "Ошибка аутентификации!" и завершает работу.

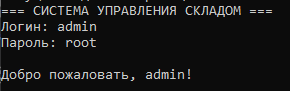


Рисунок 2 — Экран аутентификации

**2. Главное меню:**

Вывод программы: Чистый экран с нумерованным списком действий от 1 до 7 и строкой "Выберите действие:".

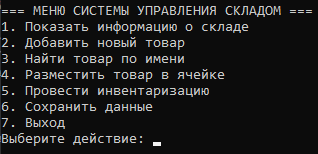


Рисунок 3 — Главное меню

**3. Пример работы функции "Поиск товара по имени" (п.3 меню):**

Вывод программы: "=== ПОИСК ТОВАРА === Введите название товара:"

Действие пользователя: Вводит часть названия, например, "Сыр".

Реакция системы: Выводит на экран список найденных товаров с полной информацией по каждому: "Найдено товаров: 1 ID: 3 | Сыр 'Российский' (Молочные продукты)..."

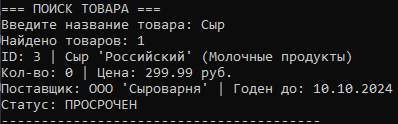


Рисунок 4 — Поиск товара по имени

**4. Пример работы функции "Инвентаризация" (п.5 меню):**

Действие пользователя: Выбирает в меню пункт 5.

Реакция системы: Выводит на экран подробный отчет по всем ячейкам склада: "=== ИНВЕНТАРИЗАЦИЯ СКЛАДА === Ячейка ID: 1 | Местоположение: Стеллаж A-1 Вместимость: 50/100 Товар: Молоко 'Простое' (ID: 1)..."

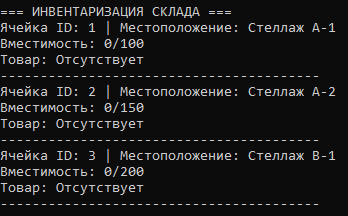


Рисунок 5 — Инвентаризация склада

**5. Пример работы функции "Сохранить данные" (п.6 меню):**

Действие пользователя: Выбирает в меню пункт 6.

Реакция системы: Пытается записать данные в файлы в папке data. При успехе выводит "Данные успешно сохранены!", при ошибке (например, нет прав) — "Ошибка сохранения: [текст ошибки]".



Рисунок 6 — Сохранение данных

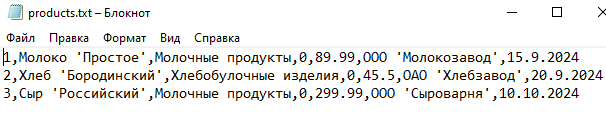


Рисунок 7 — Данные в файле

Диалог построен интуитивно понятно, каждый шаг пользователя сопровождается подсказками системы. Для перехода между этапами используется подсказка "Для продолжения нажмите Enter...", что делает интерфейс дружелюбным даже для неопытного пользователя.

##### Заключение

В ходе выполнения данной курсовой работы была успешно разработана и реализована система управления складом, отвечающая поставленным задачам. Проект демонстрирует практическое применение ключевых принципов объектно-ориентированного программирования и структурного подхода к разработке программного обеспечения.

Была создана полнофункциональная консольная программа на языке C++, позволяющая эффективно управлять товарами и ячейками хранения. Система включает в себя модули для работы с данными (добавление, поиск, размещение), проводит инвентаризацию, реализует механизм аутентификации пользователей с разными ролями и обеспечивает сохранность данных между сеансами работы через файловую систему.

Таким образом, в рамках курсовой работы была создана прочная основа для системы складского учёта. Разработанный программный продукт является полностью рабочим и готов к использованию в учебных целях, а его модульная структура и чистый код служат хорошей базой для дальнейшей модернизации и расширения функциональности в будущем.

##### Список использованной литературы

1. Курипта О.В. Основы программирования и алгоритмизации: практикум / О.В. Курипта, О.В. Минакова, Д.К. Проскурин. - Воронеж: Воронежский ГАСУ, 2015. - 132 с.

2. Неземский В.И. Процедуры и функции: методические указания / В.И. Неземский, О.А. Орешкина. - М.: Московский государственный технический университет имени Н.Э. Баумана, 2009. - 28 с.

3. Баженова И.Ю. Введение в программирование [Электронныйресурс]: учебное пособие / И.Ю. Баженова, В.А. Сухомлин. - М.: БИНОМ. Лаборатория знаний, Интернет-Университет Информационных Технологий (ИНТУИТ), 2007. - 326 с. - Режим доступа: https://intuit.ru.

4. Дейт К.Дж. Введение в системы баз данных: 7-е изд. / К.Дж.Дейт. - М.: Вильямс, 2001. - 43 с.

5. Microsoft Learn. Справочник по языку С: операторы (С): оператор switch (С) [Электронный ресурс] / Microsoft Learn: официальный сайт. - 2022. - Режим доступа: https://learn.microsoft.com

6. CppStudio. Язык программирования С++: указатели, массивы и строки: динамический массив в С++ [Электронный ресурс] / CppStudio: электронный журнал. - 2022. - Режим доступа: http://cppstudio.com/post/432/.

7.Указатели и массивы. Глава 5. Указатели и массивы [Электронный ресурс] / срр: электронный журнал. - 2022. - Режим доступа:https://learnc.info/algorithms/bubblesort.html.

8. Wikipedia. Файловый ввод/вывод в языке Си [Электронный ресурс] / Wikipedia: русскоязычная электронная библиотека. - 2022. – Режим доступа: https://ru.wikipedia.org/wiki/Файловый\_ввод-вывод\_в \_языке\_Си.

9. Солдатенко И.С. Основы программирования на языке Си: учеб.пособие. - Тверь: Твер. гос. ун-т, 2017. - 159 с.

10. Подбельский, В. В. Программирование на языке Си / В. В. Подбельский, С. С. Фомин. - Москва: Вильямс, 2004. – 600 с.

Приложение

Листинг программы на C++

#include <iostream>

#include <cstdlib> // для system()

#include <vector>

#include <string>

#include <fstream>

#include <ctime>

#include <stdexcept> // для исключений

#include <limits> // для numeric\_limits

using namespace std;

enum class UserRole {

Storekeeper,

Manager,

Administrator};

struct SimpleDate {

int day;

int month;

int year;

SimpleDate(int d = 0, int m = 0, int y = 0) : day(d), month(m), year(y) {}

string toString() const {

return to\_string(day) + "." + to\_string(month) + "." + to\_string(year);}

bool isExpired() const {

if (year < 2025) return true;

// Простая проверка: если год меньше текущего, товар просрочен

// (Для реального проекта используйте библиотеку <chrono>)

return false;}};

class Product {

private:

int id;

string name;

string category;

int amount;

double price;

int expirationDays;

string supplier;

SimpleDate receiptDate; // Используем SimpleDate

public:

Product(int id, const string& name, const string& category, int amount,

double price, int expirationDays, const string& supplier,

const SimpleDate& receiptDate = SimpleDate())

: id(id), name(name), category(category), amount(amount), price(price),

expirationDays(expirationDays), supplier(supplier), receiptDate(receiptDate) {}

int getId() const { return id; }

const string& getName() const { return name; }

const string& getCategory() const { return category; }

int getAmount() const { return amount; }

double getPrice() const { return price; }

const string& getSupplier() const { return supplier; }

const SimpleDate& getReceiptDate() const { return receiptDate; }

void setAmount(int newAmount) { amount = newAmount; }

bool isExpired() const {

return receiptDate.isExpired();}

void printInfo() const {

cout << "ID: " << id << " | " << name << " (" << category << ")" << endl;

cout << "Кол-во: " << amount << " | Цена: " << price << " руб." << endl;

cout << "Поставщик: " << supplier << " | Годен до: " << receiptDate.toString() << endl;

cout << "Статус: " << (isExpired() ? "ПРОСРОЧЕН" : "ГОДЕН") << endl;

cout << "----------------------------------------" << endl;}};

class StorageCell {

private:

int id;

string location;

int maxCapacity;

const Product\* currentProduct;

int currentAmount;

public:

StorageCell(int id, const string& location, int maxCapacity)

: id(id), location(location), maxCapacity(maxCapacity), currentProduct(nullptr), currentAmount(0) {}

void reserve(const Product\* product, int amount) {

if (amount <= 0) {

throw invalid\_argument("Количество должно быть положительным");}

if (currentAmount + amount > maxCapacity) {

throw runtime\_error("Недостаточно места в ячейке. Максимум: " + to\_string(maxCapacity));}

if (!isEmpty() && currentProduct != product) {

throw runtime\_error("Ячейка уже содержит другой товар");}

currentProduct = product;

currentAmount += amount;}

bool isEmpty() const {

return currentProduct == nullptr;}

// Геттеры

int getId() const { return id; }

int getMaxCapacity() const { return maxCapacity; }

int getCurrentAmount() const { return currentAmount; }

const Product\* getCurrentProduct() const { return currentProduct; }

const string& getLocation() const { return location; }

void printInfo() const {

cout << "Ячейка ID: " << id << " | Местоположение: " << location << endl;

cout << "Вместимость: " << currentAmount << "/" << maxCapacity << endl;

if (!isEmpty()) {

cout << "Товар: " << currentProduct->getName() << " (ID: " << currentProduct->getId() << ")" << endl;}

else { cout << "Товар: Отсутствует" << endl;}

cout << "----------------------------------------" << endl;}};

class User {

private:

string login;

string passwordHash; // В реальном проекте здесь должен быть хэш

UserRole role;

public:

User(const string& login, const string& password, UserRole role)

: login(login), passwordHash(password), role(role) {

} // Упрощенно: храним пароль открыто

bool authenticate(const string& inputLogin, const string& inputPassword) const {

return (login == inputLogin && passwordHash == inputPassword);}

UserRole getRole() const { return role; }

const string& getLogin() const { return login; }};

class Warehouse {

private:

string name;

string address;

vector<Product> products;

vector<StorageCell> cells;

public:

Warehouse(const string& name, const string& address) : name(name), address(address) {}

void addProduct(const Product& product) {

products.push\_back(product);}

void addCell(const StorageCell& cell) {

cells.push\_back(cell);}

Product& findProductById(int id) {

for (auto& product : products) {

if (product.getId() == id) {

return product;}}

throw runtime\_error("Товар с ID " + to\_string(id) + " не найден");}

// Поиск товаров по имени

vector<Product\*> findProductByName(const string& name) {

vector<Product\*> found;

for (auto& product : products) {

if (product.getName().find(name) != string::npos) {

found.push\_back(&product);}}

return found;}

void placeProduct(Product& product, int cellId, int amount) {

for (auto& cell : cells) {

if (cell.getId() == cellId) {

cell.reserve(&product, amount);

product.setAmount(product.getAmount() + amount);

return;}}

throw runtime\_error("Ячейка с ID " + to\_string(cellId) + " не найдена");}

void conductInventory() {

cout << "\n=== ИНВЕНТАРИЗАЦИЯ СКЛАДА ===" << endl;

for (const auto& cell : cells) {

cell.printInfo();}}

const vector<Product>& getAllProducts() const { return products; }

const vector<StorageCell>& getAllCells() const { return cells; }

vector<StorageCell>& getAllCells() { return cells; }

void printInfo() const {

cout << "Склад: " << name << endl;

cout << "Адрес: " << address << endl;

cout << "Количество товаров: " << products.size() << endl;

cout << "Количество ячеек: " << cells.size() << endl;}};

class FileDatabase {

private:

string basePath = "data/";

public:

void saveProducts(const vector<Product>& products) {

system(("mkdir \"" + basePath + "\" 2>nul").c\_str());

ofstream file(basePath + "products.txt");

for (const auto& product : products) {

file << product.getId() << "," << product.getName() << ","

<< product.getCategory() << "," << product.getAmount() << ","

<< product.getPrice() << "," << product.getSupplier() << ","

<< product.getReceiptDate().toString() << "\n";}

file.close();}

void saveProductCellRelations(const vector<StorageCell>& cells) {

ofstream file(basePath + "product\_cells.txt");

for (const auto& cell : cells) {

if (!cell.isEmpty()) {

file << cell.getId() << ","

<< cell.getCurrentProduct()->getId() << ","

<< cell.getCurrentAmount() << ","

<< cell.getMaxCapacity() << "\n";}}

file.close();}

vector<Product> loadProducts() {

vector<Product> products;

return products;}};

void pressEnterToContinue() {

cout << "\n[Для продолжения нажмите Enter...]";

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

cin.get();

system("cls"); // Для Windows. Для Linux/Mac: system("clear");}

void printMenu() {

cout << "\n=== МЕНЮ СИСТЕМЫ УПРАВЛЕНИЯ СКЛАДОМ ===" << endl;

cout << "1. Показать информацию о складе" << endl;

cout << "2. Добавить новый товар" << endl;

cout << "3. Найти товар по имени" << endl;

cout << "4. Разместить товар в ячейке" << endl;

cout << "5. Провести инвентаризацию" << endl;

cout << "6. Сохранить данные" << endl;

cout << "7. Выход" << endl;

cout << "Выберите действие: ";}

int main() {

system("chcp 1251");

FileDatabase db;

Warehouse myWarehouse("Основной склад", "ул. Складская, 1");

vector<User> users;

users.emplace\_back("storekeeper", "pass123", UserRole::Storekeeper);

users.emplace\_back("manager", "secret", UserRole::Manager);

users.emplace\_back("admin", "root", UserRole::Administrator);

Product milk(1, "Молоко 'Простое'", "Молочные продукты", 0, 89.99, 5, "ООО 'Молокозавод'", SimpleDate(15, 9, 2024));

Product bread(2, "Хлеб 'Бородинский'", "Хлебобулочные изделия", 0, 45.50, 3, "ОАО 'Хлебзавод'", SimpleDate(20, 9, 2024));

Product cheese(3, "Сыр 'Российский'", "Молочные продукты", 0, 299.99, 30, "ООО 'Сыроварня'", SimpleDate(10, 10, 2024));

myWarehouse.addProduct(milk);

myWarehouse.addProduct(bread);

myWarehouse.addProduct(cheese);

StorageCell cell1(1, "Стеллаж A-1", 100);

StorageCell cell2(2, "Стеллаж A-2", 150);

StorageCell cell3(3, "Стеллаж B-1", 200);

myWarehouse.addCell(cell1);

myWarehouse.addCell(cell2);

myWarehouse.addCell(cell3);

cout << "=== СИСТЕМА УПРАВЛЕНИЯ СКЛАДОМ ===" << endl;

string login, password;

cout << "Логин: "; getline(cin, login);

cout << "Пароль: "; getline(cin, password);

User\* currentUser = nullptr;

for (auto& user : users) {if (user.authenticate(login, password)) {

currentUser = &user;

break;}}

if (!currentUser) {

cout << "Ошибка аутентификации!" << endl;

pressEnterToContinue();

return 1; }

cout << "\nДобро пожаловать, " << login << "!" << endl;

int choice = 0;

while (choice != 7) {

printMenu();

cin >> choice;

cin.ignore(); // Очистка буфера

switch (choice) {

case 1: {

myWarehouse.printInfo();

pressEnterToContinue();

break;}

case 2: { cout << "\n=== ДОБАВЛЕНИЕ НОВОГО ТОВАРА ===" << endl;

// Здесь должен быть код ввода данных о товаре

cout << "Функция добавления товара реализуется здесь..." << endl;

pressEnterToContinue();

break;}

case 3: { cout << "\n=== ПОИСК ТОВАРА ===" << endl;

string searchName;

cout << "Введите название товара: "; getline(cin, searchName);

vector<Product\*> found = myWarehouse.findProductByName(searchName);

if (found.empty()) {

cout << "Товары не найдены." << endl;}

else {cout << "Найдено товаров: " << found.size() << endl;

for (const auto\* product : found) {

product->printInfo();}}

pressEnterToContinue();

break;}

case 4: { cout << "\n=== РАЗМЕЩЕНИЕ ТОВАРА В ЯЧЕЙКЕ ===" << endl;

try {Product& product = myWarehouse.findProductById(1); // Берем первый товар

myWarehouse.placeProduct(product, 1, 50); // Размещаем в первую ячейку

cout << "Товар '" << product.getName() << "' размещен в ячейке 1" << endl;}

catch (const exception& e) {

cout << "Ошибка: " << e.what() << endl;}

pressEnterToContinue();

break;}

case 5: {myWarehouse.conductInventory();

pressEnterToContinue();

break;}

case 6: { cout << "\n=== СОХРАНЕНИЕ ДАННЫХ ===" << endl;

try {db.saveProducts(myWarehouse.getAllProducts());

db.saveProductCellRelations(myWarehouse.getAllCells());

cout << "Данные успешно сохранены!" << endl;}

catch (const exception& e) {

cout << "Ошибка сохранения: " << e.what() << endl;}

pressEnterToContinue();

break;}

case 7: {

cout << "Выход из системы..." << endl;

break;}

default: {

cout << "Неверный выбор!" << endl;

pressEnterToContinue();}}}

return 0;}