ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛГИЙ И ЕСТЕСТВЕННЫХ НАУК
КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАМНОГО ОБЕСПЕЧЕНИЯ

Автоматизированная система комплектации заказов на складе предприятия

Выпускная квалификационная работа

обучающегося по направлению подготовки
02.03.03 математического обеспечения и администрирования информационных систем
очной формы обучения,
группы 07001302
Переплетчика Алексея Владимировича

Научный руководитель
доц. Муромцев В.В.

БЕЛГОРОД 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Сейчас в управлении цепочкой поставок различных грузов важная роль принадлежит логистическим центрам – многофункциональным складским комплексам, которые предназначены для работы с грузами больших объемов, завозимых разным транспортом. При поступлении в логистический центр товар должен проходить несколько этапов. Современный склад, который составляет часть логистического центра – это целый комплекс, в котором расположены технологическое оборудование, например, стеллажи. Поэтому необходимо рациональное размещение на стеллажах различных товарных единиц, а также поиск эффективного маршрута движения погрузчика по складу.

На практике фактически каждое оптовое торговое предприятие имеет свой склад, но не все из них планируют использовать на складе систему автоматизации. Это объясняется достаточно большим числом факторов. Некоторая часть из них не зависит напрямую от характеристик самого склада, а в большей степени связана со спецификой предприятия и отрасли, к которой оно относится. Среди предприятий, которые внедряют системы учета и управления складскими комплексами можно выделить следующие категории: производственные предприятия, торговые предприятия и предприятия, которые оказывают услуги ответственного хранения.

Торговые предприятия, в отличие от производственных, не обладают лишними производственными площадями. Это заставляет заниматься оптимизацией эксплуатации своих складов и тем самым увеличивать обороты товара на данной складской площадке. В это же время издержки, которые идут на содержание склада и остатков товара, являются достаточно большой статьей затрат в общей статье расходов компании.

Различные проекты автоматизации, целью которых является в основном

устранение базовых проблем склада, можно выполнить в сравнительно короткие сроки и достаточно экономично. Это зависит только от применяемой технологии идентификации товара и желаемого набора функций.

Данная задача, позволяющая управлять потоком продукции на складе, соответствует задаче многокритериальной оптимизации, в которой сначала необходимо отыскать заданную продукцию, а потом найти кратчайший маршрут сбора товара с полок в заданные зоны хранения.

Только большие предприятия могут разрешить позволить себе купить склад в собственность. Остальные организации вынуждены арендовать склады, и следовательно, для них возможна смена условий аренды, что неблагоприятно для организации. Кроме этого, смена места склада может быть вызвана ростом оборотов предприятия, а также увеличением ассортимента, сменой условий хранения и обработки продукции. Это способствует усложнению управления складом. В таких случаях введение системы автоматизации является хорошим выходом из положения. Но, необходимо заметить, что автоматизация склада во время смены места имеет свои особенности. Как правило, необходимо сделать проект в короткое время в относительно неопределенной среде (новый склад, новое оборудование, новые технологии грузопереработки и т.п.). Поэтому, чтобы обеспечить необходимую скорость переезда и быстрое расположение на новом месте, проекту автоматизации необходимо базироваться на самых простых функциях систем и технологиях работы склада. Следующим этапом идет совершенствование склада и системы.

Торговых организаций, которые планомерно решают задачи и занимаются совершенствованием склада и системы автоматизации, пока мало на российском рынке. Чтобы дать наибольшую пропускную способность склада, дать сложную аналитику, автоматизировать специфические операции, нужно ставить задачи «на перспективу», которые могут решать только крупные, устойчивые организации, располагающие достаточным для заказанного проекта бюджетом, с собственной развитой логистикой и IT-службой, способной

помогать качественному исполнению проекта автоматизации и снабжать сопровождение полнофункциональной системы управления складом.

Очень важная для автоматизации характеристика — ассортимент склада. Влияние ассортимента склада происходит непосредованно через топологию склада и технологию работы с продукцией. Торговая компания, которая «доросла» до необходимости складской системы автоматизации, имеет, значительный ассортимент товара — от нескольких сотен до нескольких десятков тысяч артикулов.

Если весь ассортимент имеет почти равный вес и габариты, условия хранения, условия отгрузки , тогда для них можно применять равные или сходные системы хранения (стеллажи, полки, консоли…), и в этом случае будут иметь место тот же самый процесс грузопереработки и логики исполнения отдельных складских операций. При этом проект автоматизации получится достаточно простым по функционалу и быстрым в исполнении.

Большое количество покупателей торговой компании и поставщиков — это один из главных факторов, которого часто не хватает у производственного предприятия.

Влияние, которое они оказывают на работоспособность склада и, следовательно, на процесс автоматизации, достаточно большое. Чем выше данные числа, тем более интенсивно работает склад, тем больше заказов в единицу времени, увеличивается каталог продукции в заказе отбираемых товаров, больше оформляется документации, меньше времени тратится на исправление ошибок и, как следствие, более высокие требования предъявляются к качеству комплектации… Продолжать список можно долго. С точки зрения системы автоматизации более важно не количество товаров, которое обрабатываются за единицу времени, а их виртуальное представление в виде количества разных бумаг, количества строк в различных бумагах, количества мест хранения склада, среди которых и происходит отбор продукции и т.п. Другими словами, выбор 10 000 единиц продукции, представленных

одной строкой задания, будет обрабатываться системой гораздо быстрее, чем 10 единиц продукции по одному товару, который находится в разных местах хранения, но, сразу видно, что это более физически трудоемко чем выполнение первого задания в отличии от второго.

Задача автоматизации склада торговой организации - это творческий процесс, который зависит от большого числа разных связанных условий. Потребности самой организации, специализация её направления (крупный опт, мелкий опт, розница…), присутствие смежных складских функций (оказание услуг ответственного хранения), условия внешней среды (работа с поставщиками) — все это влияет на процесс внедрения складской системы. Поэтому сочетание проектных испытанных технологий автоматизации и учета специфики данной организации позволяет достичь нужного результата в необходимые сроки при грамотном использовании ресурсов.

Данная ВКР рассматривает проблему автоматизации комплектации заказа на складе. В ходе написания дипломной работы были написаны 3 главы. В первой главе производится анализ задачи и постановка задачи. Во второй главе производится разработка алгоритма комплектации заказа, проектирования приложения, интерфейса пользователя и структуры БД. В третье главе показана реализация ключевых классов, библиотеки классов, реализация двух ПО и тестирование. В ней использовалось 5 таблиц и 31 рисунка

Целью ВКР являлась разработка автоматизированной системы комплектации заказов на складе предприятия, оптимизирующую перемещения комплектовщика.

Для ее выполнения необходимо решить следующие задачи:

- разработать алгоритм комплектации товаров на складе;
- обосновать выбор программных средств;
- разработать модель базы данных;
- спроектировать структуру приложения;
- спроектировать структуру интерфейса пользователя;

●реализовать в виде библиотеки классов алгоритм нахождения кратчайшего маршрута;

●реализовать ПО подготовки данных;

●реализовать ПО комплектации заказов;

●произвести проверку работы системы на тестовых заданиях.

1. ГЛАВА. ОБЗОР И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Анализ задачи комплектации заказов на складе.

Простыми методами отбора управлять легче, они легче контролируются, это приводит к тому, что при отсутствии технологической возможностей, можно использовать простые методы оправдывает себя и позволяет эффективно работать. В настоящее время стало появляться больше различных возможностей, но дискретная комплектация применяется почти везде. В какой-то мере это оправдывает себя: чем проще метод тем им проще управлять. Человек испытывает ответсвенность за ту работу которую он выполняет, он для него более понятна. Психологически это будет более комфортно.

Развитие методов «разбивает» операции, они начинают напоминать конвейер, и все более болезненно воспринимается людьми, начинает появляться «синдром конвейера». Люди не могут и не хотят с одним и тем же ритмом делать одну и ту же работу. И есть ситуации, в которых дискретный сбор актуален. При части обстоятельств - да. Если заказы почти совпадают, если отбор идет в подобной таре, например, только в ящиках. Когда время выполнения данных заказов удовлетворяет сроку выполнения одним комплектовщиком. Если путь, который комплектовщик проходит, когда собирает заказы, не превышает физиологическую норму комплектовщика. Если он не несет в руках один ящик или не везет один поддон. Существуют и другие ограничения. В том случае, когда метод эффективен, им можно пользоваться.

Но этот способ используется чаще, чем необходимо. И не используют технологические возможности, которые дают дорогие системы. Этому есть несколько объяснений.

1.Организационная проблема. Низкая квалификация на всех уровнях.

*2.*Проблема обучения. Не все поставщики занимаются обучением пользователей своих систем. Время обучения занимает очень мало времени. Диспетчеров учат значительно дольше, но недостаточно, около 7-10 дней. Это приводит к тому, что они обычно не могут оперативно использовать различные методы отбора.

Процессы, предшествующие отбору

Несмотря на распространённость дискретного метода, другие методы тоже становятся популярны. Сначала мы будем рассматривать процессы, которые в WMS предшествуют отбору. Перед началом отбора происходит резервирование продукции под заказ. По правилам, которые заложены в систему, сначала выбирают те места хранения, из которых будет отбираться продукция, и количество мест закладывается для данного заказа. В другой заказ она уже не попадет. В представленных в данное время на рынке системах есть понятие маршрута отбора. В своем большинстве маршруты жесткие и единственные. Для данного заказа будет выбрана последовательность обхода мест хранения, которые являются подмножеством заложенного маршрута. [2]

Комплектация (пикинг,подбор,сбор,) заказа

Комплектация заказа (подбор, сбор или пикинг) — это процесс забора товара с его ячейки для дальнейшей отгрузки клиенту.

Считают, что комплектация заказа — самый сложный, долгий складской процесс.

Комплектация сильно зависит от того, насколько быстро работает персонал на складе, что в свою очередь влияет на эффективность бизнес-процессов, происходящих на складе.[1]

Существует 6 видов комплектации заказов на складе:

1)дискретная комплектация (discrete picking);

2)встречная комплектация (colliding picking);

3)чередующаяся комплектация(Alternating picking);

4)комплектация партиями (batch picking);

5)зоновая комплектация (zone picking);

6)волновая комплектация (wave picking).

Дискретная комплектация (discrete picking)

Процесс сборки заказа, где один комплектовщик обрабатывает один заказ, называется дискретной комплектацией. Когда на складе используется данный метод, то комплектовщик проходит большие расстояния, и, разумеется, тратится большое количество времени на перемещения между точками отбора.

Достоинства: простота реализации, не требовательна к складу.

Недостатки: один комплектовщик работает только с одним заказом, много лишних перемещений комплектовщика.

Встречная комплектация(colliding picking)

Подключение второго комплектовщика к сбору заказа в обратном порядке называется встречной комплектацией. В этом случае два человек идут с разных концов. Этот вид комплектации выгоден там, где в основном используется дискретная комплектация. При крупных заказах можно подключить ещё одного человека.

Достоинства: простота реализации, эффективнее чем дискретная комплектация, может быть использована вместе с дискретной комплектацией.

Недостатки: требует для одного заказа больше людей.

Чередующаяся комплектация(alternating picking)

Комплектация, при которой к списку продукции подключают группу людей, в которой они по очереди получают задания, называют чередующийся комплекцией. В этом виде, в котором делят на четные и нечетные строки, один комплектовщик может имеет возможность получить несколько заданий подряд, если остальные долго выполняют свои. Метод не эффективный по количеству перемещений. В случае, при котором логика системы более развита, может быть немного улучшена система перемещений. Этот метод может быть использован в предыдущем методе для подключения третьего комплектовщика.

Достоинства: может быть использован как дополнение к встречной комплектации, быстрее чем встречная комплектация.

Недостатки: комплектовщики проходят практически весь маршрут, нерациональное использование комплектовщиков.

Комплектация партиями (batch picking)

При комплектации партиями комлектовщик собирает сразу несколько заказов. Чтобы это сделать, он выбирает похожую продукцию с их ячеек и несет их в зону комплектации, где они распределяются по заказам.

Достоинства: один комплектовщик работает с несколькими заказами, экономия времени при большом кол-ве однотипных товаров.

Недостатки: потеря времени времени при малом кол-ве однотипных товаров, необходимость сгруппировать товары, затраты времени на разбор по различным заказам, возможные ошибки при комплектации заказов.

Зоновая комплектация(zone picking)

В принципе работы рабочего используется зоновая комплектация:

1. Рабочий, ответственный за определённую зону хранения, берет заказ с предыдущей области хранения или комплектует новый заказ.
2. Выбирает по заказу товары, лежащие в его области хранения.
3. Передаёт заказ в следующую область хранения или заканчивает собирать заказ.

Заказ собирается последовательно, от одной области к другой.

Достоинства: конвейерная сборка

Недостатки: возможные простои следующих зон в случае большой нагрузки на предыдущую, требовательность к складу: необходимо наличие большей территории по сравнению с другими методами.

Волновая комплектация (wave picking)

Волновая комплектация требует перед работой подготовить склад.

Перед тем, как использовать волновой метод комплектации заказов, надо сгруппировать товары на складе в разрезе значимых характеристик.

В процессе волновой сборки pick-листов заказы комплектуются в рамках т.н. “волны”. Набор позиций, который собирают в данной области хранения, называется волной.

Волновая комплектация товара похожа на метод комплектации партиями тем, что сразу отбирается продукция для всех заказов и переносится в область комплектации. Но сбор в “волне” выполняется из определённых областей хранения, в то время как в сборе партиями продукция собирается со всех мест хранения сразу.

Достоинства: быстрая сборка.

Недостатки: требует предварительную подготовку склада, необходимость сгруппировать товары, необходим анализ заказа для правильной группировки заказов.

1.2. Анализ существующих средств автоматизации комплектации товаров на складе

Автоматизированное управление складом позволяет улучшить эффективность управления процессами, которые заранее выбраны и оптимизированы в отношении данной продукции в цепи протоков продукции для данной схемы бизнеса. Если мы определим, как необходимо работать нашему складу, мы сможем найти необходимую для управления нашим складом систему.

Обоснование проблемы автоматизации склада

Склады являются важным звеном в технологическом процессе промышленных организации, а для розничной и оптовой торговли они служат основой. По этой причине для большей эффективности работы складов на предприятиях, пытающихся опередить конкурентов, нужно использовать самые новые технологии и рабочих, которые умеют с ними работать.

Главное в оптимизации складских хозяйств – это автоматизация

связанных с ними бизнес-процессов, которые позволяют оперативно управлять запасами, понижать затраты при следующих закупках, оптимизировать использование складского места, повышать оперативность и точность учета товаров и производительность труда.

В автоматизацию складов любой организации входит комплекс таких мероприятий, как:

●оснащение складов необходимой техникой;

●разработка и последующие внедрение комплексной информационной системы, автоматизирующей деятельность склада;

●изменение в работе склада согласно новым требованиям предприятия.

Хорошо организованное складское хозяйство способствует внедрению передовых методов организации производства, ускорению оборачиваемости оборотных средств, снижению себестоимости продукции. Промышленная автоматизация уменьшает численность персонала, обслуживающего оборудование, повышает надежность и долговечность машин, дает экономию материалов, улучшает условия труда и повышает безопасность производства.

Рассмотрим существующие на российском рынке системы.

Анализ рынка систем автоматизации склада в России

На российском рынке на данный момент выделяются некоторые известные и эффективные системы автоматизации складов:

Система управления складом класса WMS

●Система управления складом RFID

●Система автоматизации склада AVACCO

●Система управления складом «1С: Логистика»

●Exceed WMS 1000

●Solvo.WMS

Идентификация продукции и ячеек в автоматическом режиме увеличивает в разы скорость и точность большинства операций. WMS-систему применяют в своей работе менеджеры и операторы. Контрольно-управленческие функции

выполняет менеджер. Оператор взаимодействует непосредственно с грузами, при этом он руководствуется заданиями, данными менеджером или системой. Работники, когда входят в систему, вводят логин и пароль. В этом случае они увидят на мониторе только те пункты, к которым имеют доступ.

Система управления складом RFID (RADIO-FREQUENCY-INDENTIFICATION)

Технология RFID Radio-frequency identification предлагает современное и оригинальное решение для учета и контроля продукции на складе.

Привычные штрих-коды раньше хорошо справлялись с задачей учета и контроля потока продукции, но в настоящее время, когда на складах хранится гигантское количество продукции, этот процесс становиться все более трудоемки и долгим. К тому же проблема безопасности и прекращения воровства остается на невысоком уровне.

Для эффективной работы склада современный рынок требует достаточно определенные условия, то есть скорость и точность потока товаров являются очень важными показателями работы склада. RFID технология позволяет решать почти все вопросы, которые стоят перед любым складом.

Недостатки и преимущество этой системы для удобства занесены в таблицу(см. таблицу 1.1):

Таблица 1.1

Достоинства и недостатки RFID

Преимущества	Недостатки
Возможность регистрации любого движения товара и отражение этого в документации.	Если метка повреждена, работа невозможна.
Требования к характеристикам ПК достаточно невысоки, и это позволяет снизить затраты на аппаратуру, что увеличивает цену/качество системы.	Самостоятельно изготовить достаточно сложно.
Адаптация к складам различного профиля.	Электромагнитные поля оказываются серьёзной помехой.

Сбор и хранение данных о каждой единице продукции	Большая стоимость этой системы по сравнению с теми же системами, базирующихся на штрих-кодах.
Улучшает работу сотрудников склада. Система уменьшает вероятность возникновения случайной ошибки и информирует работников о их задачах.	

Система автоматизации склада AVACCO

Эту систему можно использовать как полноценную систему для комплексной автоматизации склада, так как она дает возможность производить учет продукции и мониторить ее запас.

Эту систему можно адаптировать для того, что бы автоматизировать склады различной специализации (промышленной, производственной, торговой). Всегда учитывается особенность работы предприятия и учета складских операций.

При автоматизации учета продукции можно использовать технологию штрих-кода.

Данная система имеет следующие достоинства и недостатки(см. Таблицу 1.2).

Таблица 1.2

Достоинства и недостатки AVACCO

Преимущества	Недостатки
Регистрировать любое движение товара, что отражается как во внутренних приходно-расходных документах, так и в документах финансовой отчетности.	Работа при повреждении метки невозможна.

Невысокие требования к характеристикам компьютеров, что снижает затраты на оборудование и, соответственно, повышает соотношение цены/качества системы.	Сложность самостоятельного изготовления.
Система может быть адаптирована для автоматизации складов различного профиля.	Подверженность помехам в виде электромагнитных полей.
Вести сбор и хранения информации о каждом конкретном товаре .	Стоимость системы учета выше аналогичной на штрих кодах.
Повышает эффективность работы сотрудников склада.	

Система автоматизации склада «1С:логистика»

«1С:логистика» – дополнительный модуль системы (КИС) 1С. В России это решение довольно популярно. 1С является полнофункциональной линейка решений. Это решение рекомендовано среднему и малому сегменту рынка. Оно оптимально для складов с размерами до 4-3 тысяч квадратных метров со средней интенсивностью. Для него характерна возможность самостоятельного внедрения и развития при низкой стоимости. При условии, что 1С подходит вам по всем критериям, решение на базе 1С является для вас приемлемым. При выборе WMS на базе 1С следует учесть все минусы и плюсы такого решения, а также будет ли это решение отвечать всем требованиям для вашего WMS. На базе 1С логистики качество внедрения WMS сильно зависит от выбранного интегратора и его опыта решения задач разных сложностей для различных складов, а также от наличия программиста 1С с необходимым опытом работы. Последний фактор, правда, часто недооценивают. WMS на базе 1С характерно для среднего и небольшого уровня компаний, которые в качестве КИС используют 1С. Развитие и самостоятельные доработки возможны только при условии наличия в штате заказчика 1С программиста с опытом внедрения

решений WMS.(Решение для Windows/MSSQL, Продукт разработан на Visual C++). [3] (см. таблицу 1.3)

Таблица 1.3

Достоинства и недостатки «1С:логистика»

Преимущества	Недостатки
Имеется набор инструментов для доработки системы под нужды заказчика	Для хорошего внедрения нужен программист 1С
Одна из старейших систем на этом рынке и потому обладает большим опытом по внедрению.	Максимальная эффективность только при радитерминальной технологи
Поддержка штрих-кодов	Самостоятельные доработки невозможны без программистов 1С

Таблица 1.3 продолжение

Поддерживает два вида выдачи задач, бумажной и радиотерминальной технологии.	Качество внедрения очень сильно зависит от выбранного интегратора
Простота адаптации практически к любому складу	

Система автоматизации склада Exceed WMS 1000

Это решение подходит для оптимальной работы складов средних и мелких организаций. Также оно может удовлетворять требования при небольших вложениях денег во внедрение системы.

Эта система может контролировать все процессы, которые связаны с дистрибуцией и пополнением запасов, что снижает траты на обработку и хранение продукции.

EXceed WMS 1000 дает высокую эффективность управления складом,

включает в себя управление процессами получения, заказами, резервирования, пополнения запасов, хранения, комплектации, отбора, отчетности и отгрузки.

Имеет следующие достоинства и недостатки:(см таблицу 1.4)

Таблица 1.4

Достоинства и недостатки Exceed WMS 1000

Преимущества	Недостатки
Поддержка 24 языков	Подходит только для фармацевтики и автозапчастей
Поддержка процесса пикинга с динамическими или фиксированными ячейками	Закрыт базовый функционал
Наличие средств обеспечения безопасности	Высокая стоимость доработок
Графический пользовательский интерфейс	
Богатые встроенные возможности формирования запросов	

Система автоматизации склада Solvo.WMS

Solvo.WMS — это складская комплексная автоматизированная система, которая обеспечивает решение задачи по автоматизации склада и процессами, происходящими на нем. Solvo.WMS принадлежит к классу программно-аппаратных систем управления складами. Solvo.WMS направлена на большую область применений и может эффективно работать на складах.

Solvo.WMS является одной из экспертных систем, которые способны сами создавать рекомендации по оптимальной работе всех складских технологических процессов для достижения наибольшей успешности использования складских помещений и поднятия производительности труда.

Благодаря данной системе пользователь может не составлять описания выборки и заказа на бумаге. Они обычно составляются главным компьютером

или человеком. Вместо этого нужная информация обрабатывается и передается самой системой непосредственно, затем она преобразуется в эффективные задания для каждого комплектовщика.

Каждая операция сразу же вводится в систему с помощью клавиатуры или сканированием. Это говорит о том, что данные о расположении и количестве товара на складе всегда актуальны и точны и любые отклонения учитываются немедленно.

Также в задачи данной системы включается управление всеми складскими процессами от приемки продукции до отгрузки, а затем и доставки. Исходя из реальных нужд клиента, объем внедрения системы может меняться от базового уровня до полноценной автоматизированной системы управления складом в режиме реального времени, которая будет использовать технологии штрих-кода и различных средств автоматизации.

Данная автоматизированная система позволяет эффективно хранить товар, разделять склад на области так, чтобы площади склада использовались наиболее оптимально.

Внедрение системы показало, что точность информации о размещении и количестве товара при проверке достигало 99,9%, при этом производительность труда у складских работников повышалась на 20-30%.

Важным качеством данной системы является её способность адаптироваться к условиям конкретного клиента, его особенностям, а также к технологическим и организационным требованиям. ПО может работать со штрих-кодом, принтерами, сканерами, электронными весами.

К недостаткам данной системы можно отнести высокую стоимость самой системы, услуг внедрения, её сопровождения и поддержки, а также невозможность доработки заказчиком.

После анализа всех описанных выше систем разберем основные критерии и впишем в таблицу данные по этим системам(см таблицу 1.5).

Таблица 1.5

Сводная таблица характеристик систем

Критерии:	RFID	AVACCO	«Логистика 1С»	Exceed WMS 1000	Solvo.WMS
Стоимость	Высокая	Высокая	Средняя	Высокая	Высокая
нужны «метоки»	Да	Да	Нет	Нет	Нет
Отслеживание товара	Да	Да	Да	Да	Да
Более одного языка	Нет	Нет	Нет	Да	Нет
Класс	Адаптируемая	Интегрируемая	Модуль КИС 1С	Конфигурируемая	Конфигурируемая
Доработка заказчиком	Возможна	Возможна	Возможна	Частичная	Невозможна
Отраслевые решения	Любая отрасль	Любая отрасль	Торговая пищевая промышленность	Фармацевтика, автозапчасти	Торговая, пищевая, формацевтика
Рейтинги	Мировой топ	Российский топ	Российский топ	Мировой топ	Российский топ
Платформа	Нет платформы	Нет платформы	1С логистика	Infor	Нет платформы
Технологии	В зависимости от системы	Ms sql, Xml, Asp, Cashe	.Net for windows IIS MsSql 3-уровневая архитектура	.ADO for Windows, Mysql, 3-уровневая архитектура	Linux, oracle, 3-уровневая архитектура

1.3. Постановка цели и задач

Анализируя данную выше таблицу (см. таблицу 1.5) приходим к выводу, что на рынке нет обеспечения автоматизации управления складом малых предприятий, которым не нужна высокая автоматизация или которые не имеют средств для закупки оборудования, необходимое для среднего и высокого уровня автоматизации. А «1С логистика», у которой есть недорогие варианты для интеграции и модернизации, требует программиста 1С, а также желательно, чтобы WMS была на базе тоже 1С. Также все рассмотренные системы автоматизации предполагают, что у склада есть возможность поставить штрих-

код или метку на товар. Но некоторые товары, чтобы поставить на них штрих-код, требуют дополнительных затрат на упаковку. А оснащать товары меткой также довольно дорогой процесс. Также стоит учесть, что необходимо купить их систему, которая не всегда «по карману» малым и средним предприятиям.

Исходя из сказанного выше, программного обеспечения не хватает именно малым и средним предприятиям, которым требуется небольшая автоматизация. С учетом того, что данное ПО будет проектироваться для небольших предприятий, стоит учесть, что склады не являются очень большими и они не имеют большого количества комплектовщиков. Это значит, что зоновая комплектация, а также волновая комплектация для них не подойдут, так как для зоновой комплектации необходимы повышенные размеры склада по сравнению с остальными, а у волновой необходимо оборудование, которое будет рассчитывать, какой товар куда класть. К тому же волновая комплектация хорошо показывает себя на больших складах, а не на малых и средних складах, где её успехи немногим выше остальных методов. Коплектация чередующаяся не подходит для малых предприятий по причине требовательности к количеству комплектовщиков. Метод партиями хорошо себя показывает тогда, когда можно правильно сгруппировать товары, что не всегда получается. Поэтому его тоже пропускаем. Подойдет же метод дискретной комплектации. Но поскольку мы имеем малое количество комплектовщиков, грузчикам необходимо предоставить возможность собирать несколько заказов сразу, проходя при этом миниимальный путь, чтобы значительно уменьшить время, затрачиваемое на сбор заказов. Поскольку в большинстве ситуаций у малых и средних складов несколько заказов комплектовщик может забрать за раз, то грузоподъемность пока учитывать не будем.

В итоге получаем, что нам необходимо добавить к методу дискретной комплектации с учетом возможности дополнения его встречной комплектацией поиск кратчайшего пути и выбор сразу нескольких заказов.

Цель дипломной работы – разработать автоматизированную систему

комплектации заказов на складе предприятия, оптимизирующую перемещения комплектовщика.

Для выполнения этой цели необходимо решить следующие задачи:

●разработать алгоритм комплектации товаров на складе;

●обосновать выбор программных средств;

●разработать модель базы данных;

●спроектировать структуру приложения;

●спроектировать структуру интерфейса пользователя;

●реализовать в виде библиотеки классов алгоритм нахождения кратчайшего маршрута;

●реализовать ПО подготовки данных;

●реализовать ПО комплектации заказов;

●произвести проверку работы системы на тестовых заданиях.

2. ГЛАВА. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ КОПЛЕКТАЦИИ ЗАКАЗОВ НА СКАДЕ

2.1. Разработка алгоритма комплектации заказов

Как определено выше, лучшим вариантом комлпектации заказов будет модернизация дискретного метода с учётом возможности выбора нескольких заказов с возможностью подключения случая встречной комплектации. Из этого выходит, что мы должны иметь на вход нашего метода следующие данные: начальная точка (begin), конечная точка (end), массив заказов (orders), склад (stock).

В случае использования встречной комплектации первый комплектовщик просто выбирает начальную точку, а другой - конечную.

1.Для начала необходимо определить хватает ли продукции на складе для данного заказа. Если нет, выдать то, что не хватает.

2.В случае, если продукции хватает, необходимо определить, каким образом мы можем собрать их со шкафов.

3.Определим для каждого из полученных вариант сбора книг минимальный путь с учетом того, что у нас для всех случаев точка начала и конца неизменна, а порядок обхода необходимых шкафов может меняться для их сбора. На основе полученных данных выберем вариант с минимальным путем.

Так как 3 пункт может занимать много времени, если у нас имеется много различных вариантов сбора и один цикл поиска минимального пути для одного варианта сбора не зависит от другого варианта сбора. В этом случае необходимо распараллелить алгоритм для ускорения работы программы. Можно будет уменьшить время работы, если заранее обработать граф, чтобы получить расстояние от любой одной точки до любой другой точки. Но необходимо будет

сохранить пути, по которым от одной точки мы достигаем другой.

В итоге получается следующая блок-схема:(см. рис. 2.1)

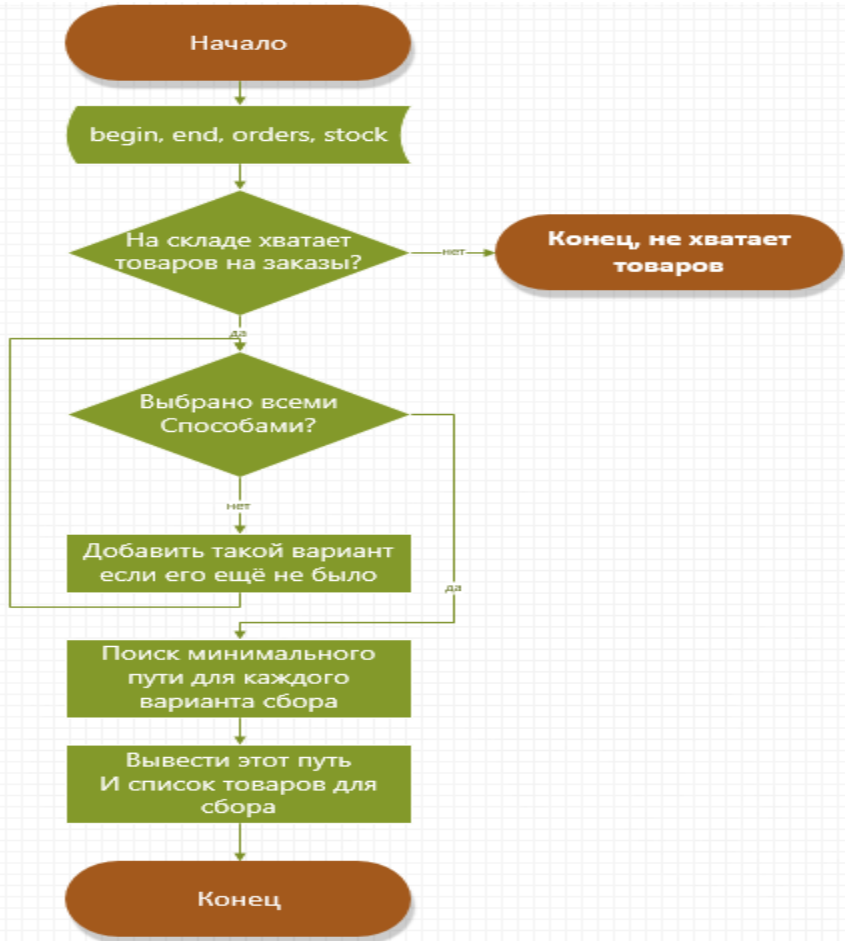


Рис. 2.1. Блок схема

Сначала необходимо из массива заказов получить данные о том, какой товар требуется и сколько. Для простоты выборки этих данных определимся, что данные сначала идут по ид товара, затем по ид заказа.

После того, как определили эти данные, необходимо получить сведения, где хранится необходимый товар. Будем определять для каждого товара в отдельности возможные способы сбора, а после этого соединять с теми местами, где нужно собрать остальной товар. Дублирующиеся места будем удалять. Для того, чтобы по каждому товару обработать возможные варианты сбора, будем использовать алгоритм порождения размещений и отнимать от необходимого количества товаров количество товара на первом стеллаже в

размещении. Если недостаточно, то отнимать от второго, и так, пока не будет достаточно. Если после сбора со всех мест товара будет недостаточно, будет выдана ошибка о том, что его недостаточно на складе. В случае, если товара достаточно, то к уже имеющимся возможным сборам мы добавим возможные сборы этого товара.

После того, как выяснены необходимые точки прохождения, нужно найти для этих методов сбора минимальный маршрут. Для ускорения этого метода заранее обработаем граф для ускорения поиска минимального маршрута.

Для обработки графа заранее будет использован алгоритм Флойда с сохранением путей, так как он дает все необходимые данные для дальнейшей работы. [6]

После этого для обработок всех вариантов обхода будем использовать алгоритм порождения всех сочетаний. Первым сочетанием будет: [0;1;2;3;4;5;6;0]. Получается следующий следующий алгоритм:

1.Подсчитать сумму расстояний по маршруту.

2.Проверить, является ли она меньше текущего минимума. Если да, то записать это как минимальный маршрут и как минимальный путь.

3.Перейти к следующему сочетанию. Если номер этого сочетания меньше, чем $(amount-1)^2+amount-1$ (где amount количество точек, которые необходимо посетить), то возвращаемся к пункту 1. Если больше, переходим к пункту 5.

4.Вернуть минимальный маршрут и его вес.

Но так как мы будем иметь дело с некоторым количеством заказов, то и точек мы будем иметь больше чем 5, что позволит его немного оптимизировать. Введя левую и правую сумму, можно оптимизировать подсчет суммы, который даст нам то, что, вне зависимости от количества точек необходимых для посещения, будет одинаковое количество действий для подсчета суммы маршрута.

Определимся, что число будет двигаться слева направо. В таком случае

левая сумма при инициализации будет считаться как вес от точки начала до первой точки. А правая сумма будет считаться как вес маршрута от второй точки до финальной. Начальное положение сумм выглядит так: левая сумма [0;1], правая часть [2;3;4;5;6;0], массив вершин [0;1;2;3;4;5;6;0].

Из-за этого итоговая сумма будет считаться как left_sum+graph[vertex[c]][vertex[c+1]]+right_sum, где left_sum — левая сумма, graph — матрица весов,vertex — массив вершин, c-номер вершины, которую будут двигать, right_sum — правая сумма.

Также при сдвиге на следующий разряд необходимо перед непосредственно сдвигом для левой суммы вычесть graph[vertex[c - 1]][c]. А для правой суммы вычесть graph[vertex[c + 1]][vertex[c + 2]].Эти вычитания делаются для отсоединения точек от маршрута перед сдвигом. Суммы в этот момент будут выглядеть так: левая сумма [], не входящие ни в одну сумму точки [1;2], правая сумма [3;4;5;6;0], массив вершин [0;1;2;3;4;5;6;0].

После сдвига необходимо к левой сумме прибавить graph[vertex[c - 1]][c] + graph[vertex[c]][vertex[c + 1]]. Это сделано для того, чтобы после сдвига их соединить с левой частью. После этого преобразования будет выглядеть следующим образом: левая сумма [0;2;1] правая сумма [3;4;5;6;0] массив вершин [0;2;1;3;4;5;6;0].

При «c» соответствующему предпоследней вершине в заказе правую часть надо преобразовать левую. Это необходимо для того, чтобы ещё раз не пересчитывать суммы в начале в следующей итерации. Перед сдвигом делаем то же, что и всегда, кроме того, что правую часть не пересчитываем. В этот момент состояние следующее: левая сумма [0;2;3;4;5] не входят [1;6;0] правая сумма [] массив вершин [0;2;3;4;5;1;6;0]. После сдвига присоединяем как и всегда две точки к левой части и получаем состояние: левая сумма [0;2;3;4;5;6;1] правая сумма [] массив вершин [0;2;3;4;5;6;1;0].

Потом отсекаем от левой части начальную точку и следующую за ней точку, отнимая от левой суммы graph[begin][vertex[0]]

+graph[vertex[0]][vertex[1]]. Получается следующие состояние: не входят [0;2], левая сумма [3;4;5;6;1], не входит [0], массив [0;2;3;4;5;6;1;0].

Следующим действием присваиваем в правую часть и присоединяем конечную вершину прибавляя graph[vertex[count-1]][end], где count - количество вершин в массиве vertex, и под конец приравниваем левую часть к той сумме, которую мы перед этим из неё вычитали. Получаем следующее состояние: левая сумма [0;2] правая часть [0;2;] массив вершин [0;2;3;4;5;6;1;0].

При «c» равном 0 стандартно вычитаем из правой вершины graph[vertex[c + 1]][number[c + 2]], после делаем сдвиг и левой сумме присваиваем graph[begin][vertex[c]] +graph[vertex[c]][vertex[c + 1]].

2.2. Обоснование выбора программных средств

2.2.1. Выбор языка программирования

Языки программирования делятся на три главных уровня: низкий, средний и высокий уровни. Поэтому определим необходимый уровень языка для нашей задачи.

Низкий уровень

Среди характеристик часто встречаются ограничения на абстракции данных, сильная статическая типизация, отсутствие промежуточной среды выполнения, прямой доступ к памяти. Их достоинства и недостатки:

Достоинства:

●Полный контроль практически над всем; вы используете только то, что вам нужно.

●Больший контроль над памятью, вы можете сделать то, что практически невозможно в других языках.

●Позволяет добиться большей эффективности.

Недостатки:

●Дополнительный контроль означает дополнительные сложности, которые могут сделать вроде бы простые задачи более трудными в реализации.

●Управление памятью может очень усложнится.

●Требуется предварительная оптимизация.

●Изменения в плохой архитектуре могут быть болезненными. А хорошую архитектуру тяжело придумать.

●Относительно бедная стандартная библиотека означает, что вы должны часто полагаться на третьих лиц или "изобретать колесо".

●Необходимо часто вставлять вспомогательные куски кода (boilerplate), что увеличивает время на разработку.

Средний уровень

Среди характеристик часто встречаются: фокус на абстракциях, сильная статическая типизация, среда выполнения, ограничения на прямой доступ к памяти.

Достоинства:

●Управлять памятью необязательно, но при желании вы можете это делать самостоятельно.

●Богатые стандартные библиотеки.

●Компилируется в байт-код, упрощающий взаимодействие с другими языками.

Недостатки:

●Байт-код требует установленной среды выполнения.

●До сих пор нужно часто вставлять стандартные куски кода (boilerplate), несмотря на наличие абстракций.

Высокий уровень

Среди характеристик часто встречаются сильное абстрагирование, динамическая и/или слабая типизация, полностью независимое управление памятью и/или наличие среды выполнения.

Достоинства:

●Абстракции делают сложные задачи простыми;

●Меньше вставок стандартного кода (boilerplate) – синтаксис значительно проще;

●В целом всё просто и интуитивно даже при внесении изменений;

●Сравнительно большие стандартные библиотеки означают, что то, что вы хотите сделать, скорее всего уже реализовано и доступно.

Недостатки:

●Надстройки для реализации абстракций снижают производительность;

●Архитектура может страдать, т.к. довольно просто вносить изменения почти в любом месте вместо того, чтобы вносить их там, где действительно нужно.

●Из-за скрытых деталей сложно выяснять причины возникновения проблем, когда они появляются.

●Динамическая типизация усложняет поиск ошибок без запуска кода.

В качестве языка программирования был выбран язык C# среднего уровня за классические для этого уровня достоинства. По сравнению с другими языками среднего уровня он имеет преимущества в виде хорошо написанных стандартизированных библиотеках. Net framework, что значительно ускоряет процесс разработки, особенно если выбрать Microsoft Visual Studio. Также стоит отменить, что из-за частичной абстракции удобнее работать с графикой. И к тому же в C# довольно удобно сделан вызов функций из других языков.

2.2.2. Выбор системы управления базами данных

Я выбрал систему Microsoft server, потому что я её знаю и поэтому разработка будет идти быстрее. При этом она обладает следующими достоинствами:

Масштабируемость:

●Алгоритмы использования дискового пространства.

Sql Server легко маштабируется от ПК до мультипроцессорных кластеров под управлением Windows NT.

●Усовершенствованный процессор запросов.

Sql Server имеет в себе новый процессор запросов который дает поддержку БД больших размеров и работу со сложными запросами и др. Среди главных новинок в нем использование составных индексов, множественные триггеры, новые алгоритмы хеширования и обработка гетерогенных, распределенных и параллельных запросов.

Превосходная производительность:

●Увеличенный размер страниц

Увеличенный до 8 КБ размер страниц способствует быстрому извлечению данных, позволяет использовать строки и столбцы большего размера, что открывает возможность эффективного хранения сложных, подробных данных.

●Динамическое блокирование на уровне строк

Менеджер блокировок динамически адаптирует алгоритм использования ресурсов в больших базах данных, что делает продукт наиболее пригодным для интерактивной обработки транзакций (online transaction processing - OLTP) и создания хранилищ данных.

Простота использования:

●Динамическое самоадминистрирование

Выполнение многих рутинных задач администрирования теперь автоматизировано.

●Средства профилирования и настройки

Средства упрощения поиска и решения проблем, происходящих во время работами с базами данных при помощи регистрирования и воспроизведения работы сервера. [7]

Существование коммерческой и ознакомительной версии:

Существует бесплатная версия express, которая от платной версии отличается лишь тем, во сколько максимум может весить база данных.

2.3.Разработка модели базы данных и структуры приложения

База данных для складского приложения - одна из главнейших частей приложения, так как в каком-либо другом виде хранить данные по складу неудобно. К тому же на ваши плечи ляжет задача одновременного доступа с различных терминалов.

При проектировании базы данных за основу были взяты три сущности: склад, заказ, ребра.

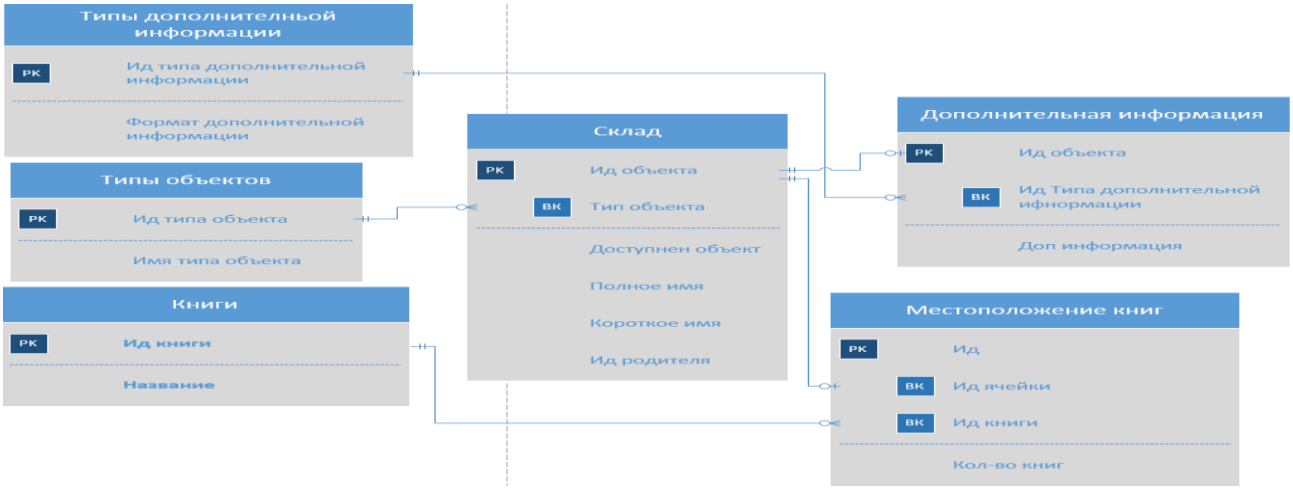
База имеет 14 таблиц, число которых увеличивается с расширением функционала. Часть из этого это справачники. Вокруг трех указанных выше таблиц строится логика проектирования базы. Список таблиц и описание их представлено ниже в виде таблицы (см. таблицу 2.1).

Таблица 2.1

Описания таблиц

Название таблицы	Описание
Additional_Info	Дополнительная информация к описанию шкафа(местоположение, кол-во ярусов, кол-во ячеек в ярусе и др.)
Additional_Types	Форматы хранения дополнительных данных
Book_Location	Описывает в какой ячейке хранятся какие книги
Books	Описание книг
Content_Order	Содержимое заказа
Lines_Ways	Содержит точки ребра
Members	Содержит информацию о заказчиках
Orders	Содержит информацию о заказе
Prices	Справочник цен по книгам в зависимости от типа заказа и др.
Rules	Справочник правил товаров
Statuses_Order	Справочник по статусам заказа
Stock	Содержит описание шкафов, полок, ярусов, ячеек а также стен и точек начала/конца
Types_Orders	Типы заказов(простой , срочной и др.)
Ways	Содержит список ребе.

Вокруг каждой из 3 ключевых таблиц образовалась группа таблиц. В первой группе собраны таблицы, отображающие данные о физических данных



объектов на складе а именно что за объект (склад), физические данные объекта (дополнительные данные), как они храняться (типы дополнительной информации) типы объектов склада, и на какой ячейке какая книга хранится и справочники к этим объектам(см. рис. 2.2).

Рис. 2.2. Представление склада и находящегося в нем

Вторая группа таблиц посвящена заказу и всему связанных с ним сущностей таких как справочники: заказчики, типы заказов, адреса, а также таблиц отображающей содержимое заказа и книг в ней, а также дополнительные таблице а о типе заказа и по какому правилу идут товары в книге. И дополнительных (см. рис. 2.3).

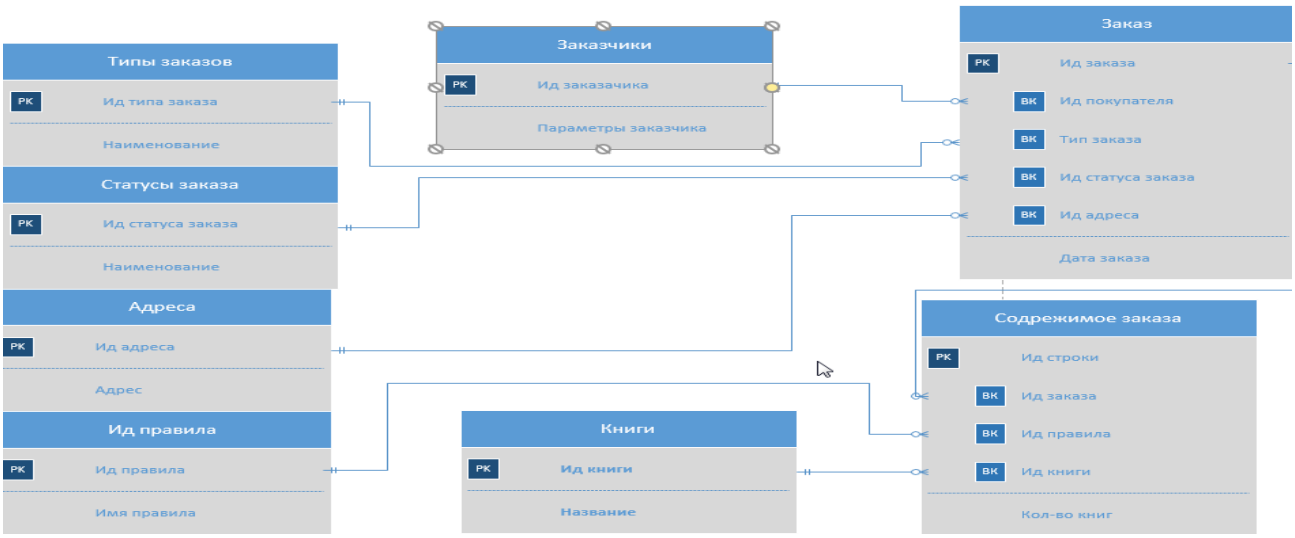


Рис. 2.3. Заказ и связанное с ним

Третья группа посвящена возможным перемещениям по складу а именно описание ребер, описание точек ребра и связанных с этими ребрами объектов склада (см рис. 2.4).

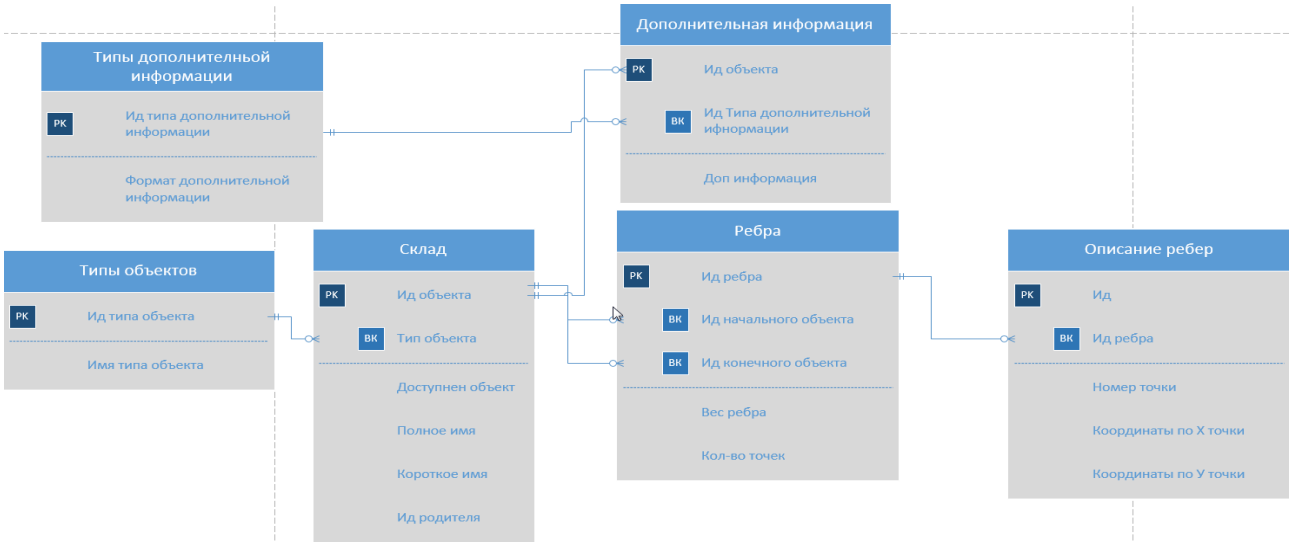


Рис 2.4. Ребра и их связь с физическими объектами

После проектирования БД можно приступить к проектированию приложения которое будет взаимодействовать с этой БД будет разбито на следующие составляющие:

1.База данных — содержит основные данные, необходимые для работы

приложения.

2.Библиотека классов

А.Класс, отвечающий за соединение с бд и операция с ним.

В.Класс, обрабатывающий граф и производящий в нем поиск кратчайшего маршрута, проходящего через заданные точки.

3.ПО подготовки данных — нужно для заполнения, обновления и удаления части данных, а также помогает с заполнением тестовых заданий.

4.ПО обработки заказов — работает с заказами: выдает кратчайший маршрут в графическом виде с возможностью его печати, меняет статусы заказа, позволяет отмечать в базе, что посылка собрана.

Для удобства отображения приложение было разбито на 3 состовляющие данные, библиотека классов и последнее собственно различные ПО для работы с этими данными. Составляющие проекта соединены следующим образом(см. рис. 2.5):

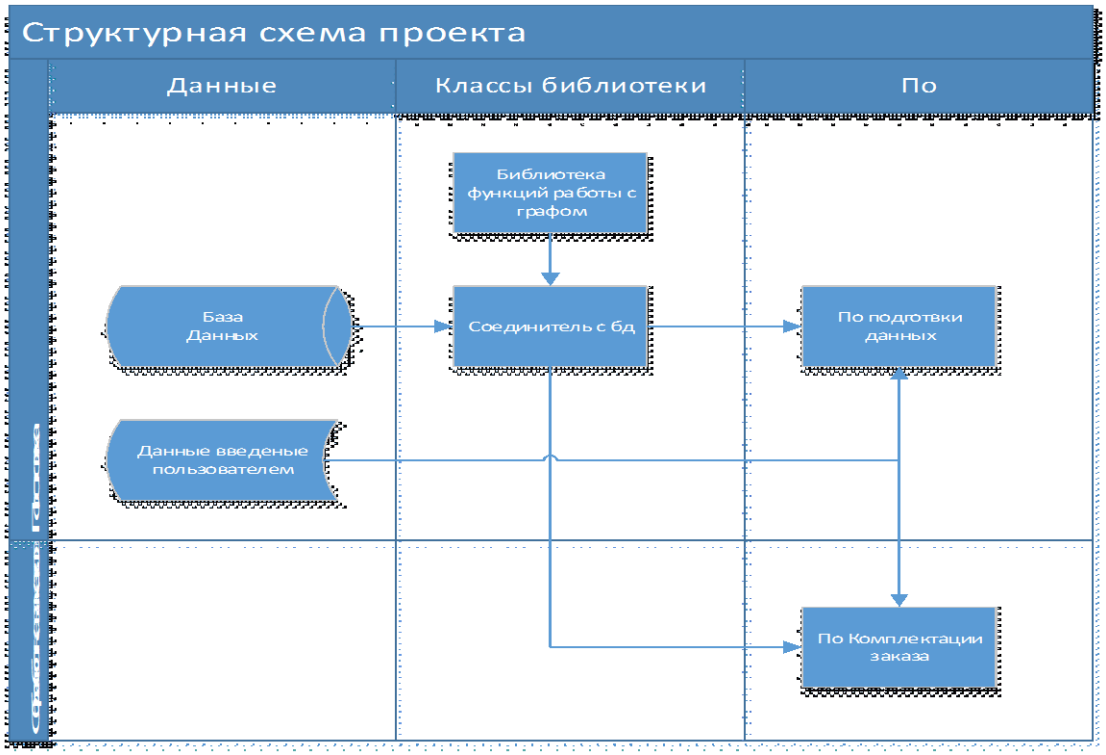


Рис. 2.5. Структурная схема приложения.

2.4. Проектирование структуры интерфейса пользователя

Первое ПО отвечает за подготовку данных. а это значит, что в нем должны быть следующие функции:

1) размещение стеллажа, стены, точка входа/выхода;

2)возможность проложить путь от одного объекта до другого (кроме стен);

3)возможность сохранить и загрузить из базы склад;

4)при случае, если изменение плана не завершено, возможность сохранить это в файл, и после загрузить из него;

5)просмотр содержимого стеллажа: сколько у него ярусов, ячеек, что лежит на ячейках, при необходимости их добавление или изменение;

Определим, какие должны быть пункты главного меню и какие подпункты.

1.Файл:

1)сохранить — с диалоговым окном сохранения;

2)загрузить — с диалоговым окном загрузки.

2.БД:

1)соединение с БД -ввести логин и пароль, БД, к которой подключаемся, и возможность выбрать настоечный файл;

2)выгрузить из БД;

3)сохранить в БД.

3.Новый склад должен выдавать форму с возможностью ввода ширины и высоты «нового» склада.

4.Очистить — удалить все объекты склада и пути на нем.

5.Добавить дизайнером — позволяет первым кликом начать рисовать объект склада, вторым закончить его добавление:

1)шкафа;

2)точки входа/выхода;

3)стены.

6.Скрыть/показать пути — скрывает или показывает уже добавленные пути на складе.

1)показать/скрыть путь из [откуда] в [куда] вес пути: [вес пути](путь первый);

А) удалить этот путь.

2)показать/скрыть путь из [откуда] в [куда] вес пути: [вес пути](путь второй);

А) удалить этот путь.

3)показать/скрыть путь из [откуда] в [куда] вес пути: [вес пути](путь n);

А) удалить этот путь.

Касательно шестого пункта так как после определенного количества следующие пункты пойдут во вторым столбиком, то это будет вполне удобно. При необходимости может быть легко переделано. Исходя из данных опции получается такая структура меню(см. рис. 2.6):

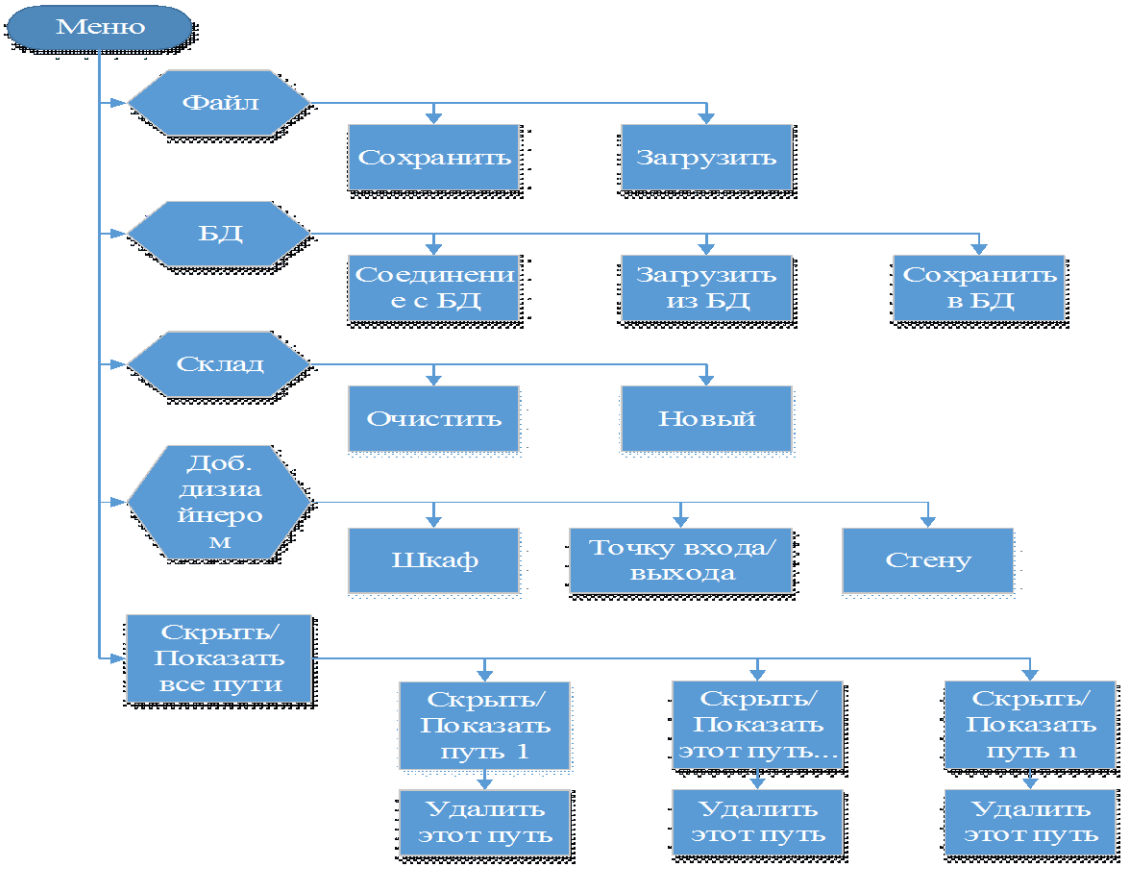


Рис. 2.6. Структура главного меню

Что касается дополнительного меню, оно необходимо для дополнительных данных по объектам и их точного добавления по координатам, и будет состоять из вкладок. Первая вкладка будет иметь возможности выбора: местоположения верней левой точки при помощи X и У и второй точки при помощи выбора его ширины и высоты. Далее следует написать полное имя стеллажа. Потом нужно выбрать, с какой стороны с него можно забирать товары, и выйдет структура, изображенная ниже(см. рис. 2.7).

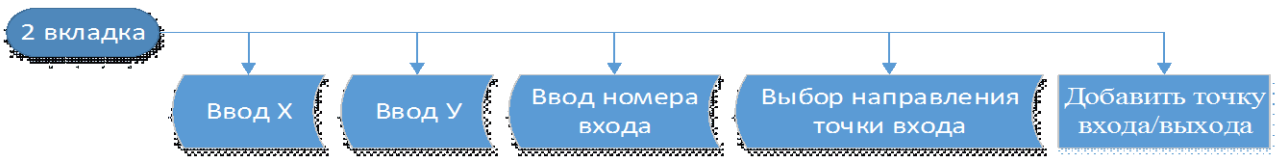


Рис 2.7. Структура 1-ой вкладки

Вторая вкладка будет иметь только пункты выбора X, У, номер входа и

направление, которое предполагает от какой точки входа/выхода и куда будет идти комплектовщик. В итоге выходит структура вкладки, представленная ниже(см. рис. 2.8).

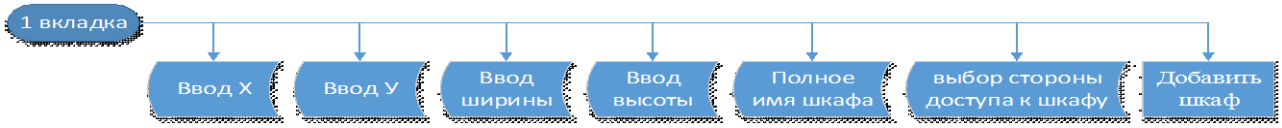


Рис 2.8. Структура 2-ой вкладки

Третья вкладка будет иметь те же пункты: X, У, ширину и высоту, которые позволят добавить стену на форму. Таким образом, выйдет структура, изображенная ниже(см. рис. 2.9).

Рис 2.9. Структура 3-ей вкладки

Нажатие на шкаф на форме правой кнопкой мыши будет удалять шкаф и его содержимое. Нажатие же левой кнопкой мыши будет давать при подключении к базе данных и условии, что этот шкаф уже есть в ней, окно выбора количества ярусов, его доступность и возможность посмотреть, что есть на каждой ячейке(см. рис. 2.10):

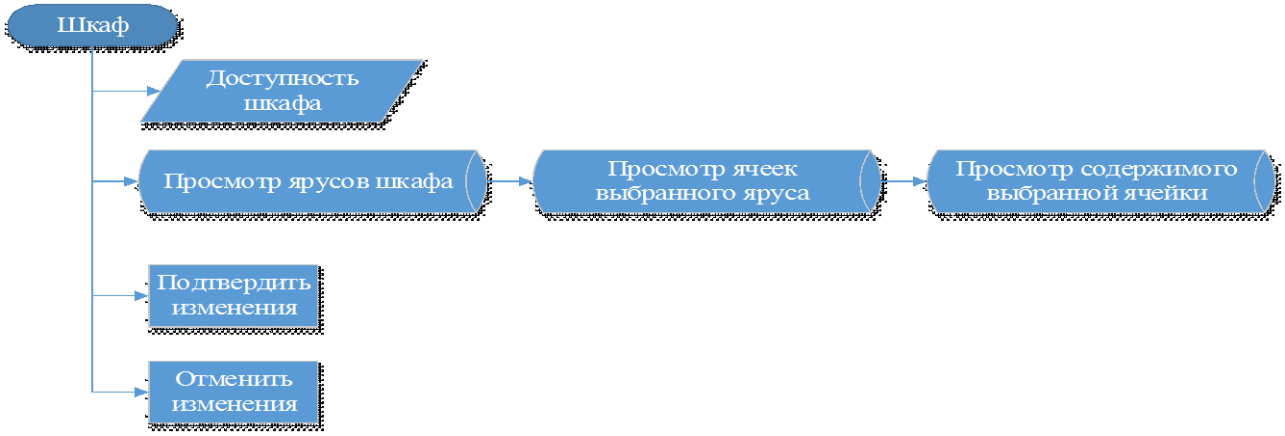


Рис. 2.10. Структура настройки и просмотра шкафа

Исходя из описанных выше вещей, которые должны быть на форме

осмотра данных шкафа(стеллажа), выходит форма (см. рис. 2.11):

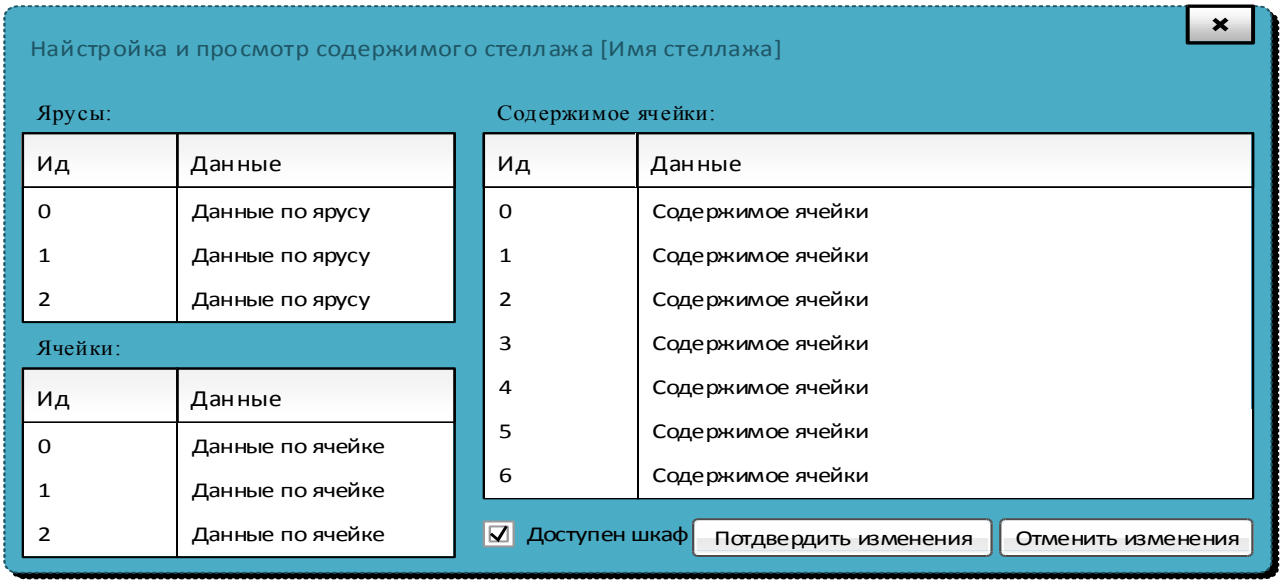
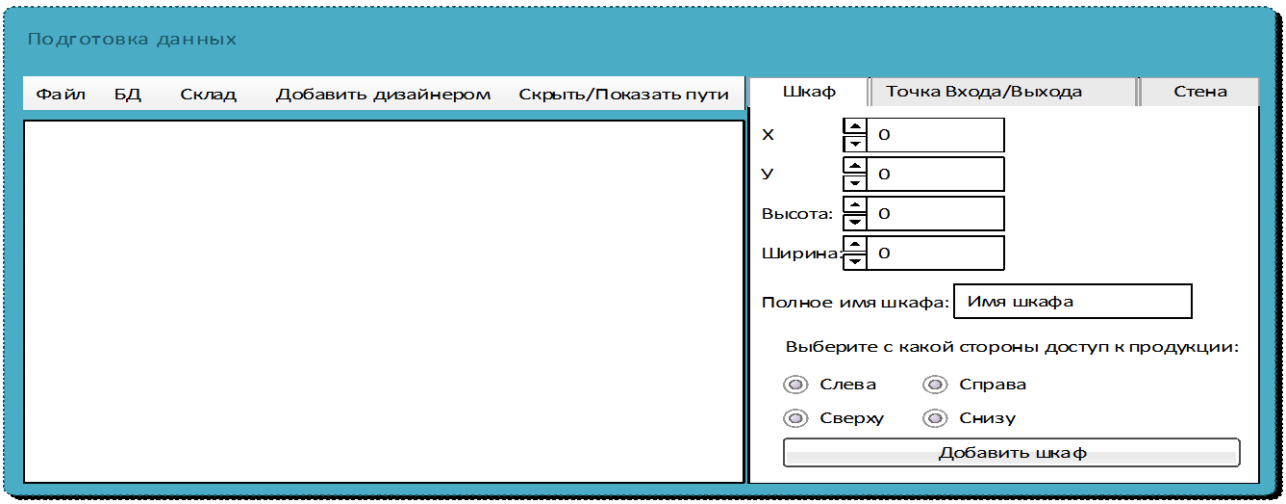


Рис. 2.11. Форма просмотра и изменений данных стеллажа

Нажатие на точку подхода или входа будет приводить к началу создания пути, каждое нажатие на форме на что-то, кроме точек подхода или входа, будет создавать новую точку в пути, повторное нажатие на них будет заканчивать путь. Если подобный путь уже существует, то больший будет



удаляться. В итоге спроектирована следующая форма(см. рис. 2.12):

Рис 2.12. форма подготовки данных

Второе же ПО комплектации товаров нужно для работы непосредственно

с заказами, а значит должно иметь следующие функции:

1)возможность просмотра заказов и их содержимого;

2)возможность выбора необходимых заказов;

3)указание для маршрута начальной и конечной точки;

4)возможность резервирования товара под выбранные заказы;

5)возможность показа кратчайшего маршрута для сбора заказа;

6)возможность перевода статуса заказа.

Определимся, какие пункты меню должны быть:

1.Вход — дает форму, в которую вводятся логин, пароль, выбирается настоечный файл и вводится строка подключения к БД.

2.Загрузить/обновить таблицы — загружает или перезагружает заказы и их содержимое, сбрасывает выделение заказов, а также загружает из базы склад.

3.Начальная точка — помечает начальную точку нажатием на форме на подход к шкафу или точку входа.

4.Конечная точка — помечает конечную точку нажатием на форме на подход к шкафу или точку входа.

5.Показать путь — высчитает кратчайший путь для сбора выбранных заказов.

6.Зарезервировать товар — на основе высчитанного пути и полученных данных резервирует товар на полках.

7.Отменить резервирование товара — может возникнуть необходимость отменить резервирование, но это должно быть доступно только на определенных статусах заказа.

Исходя из данного была получена следующая структурная схема меню ПО обработки заказов(см. рис. 2.13):

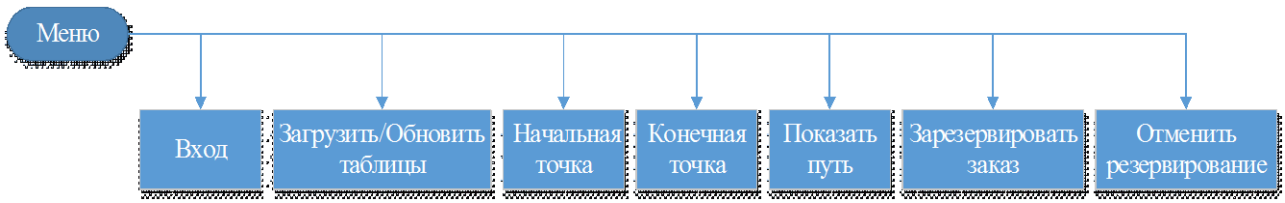
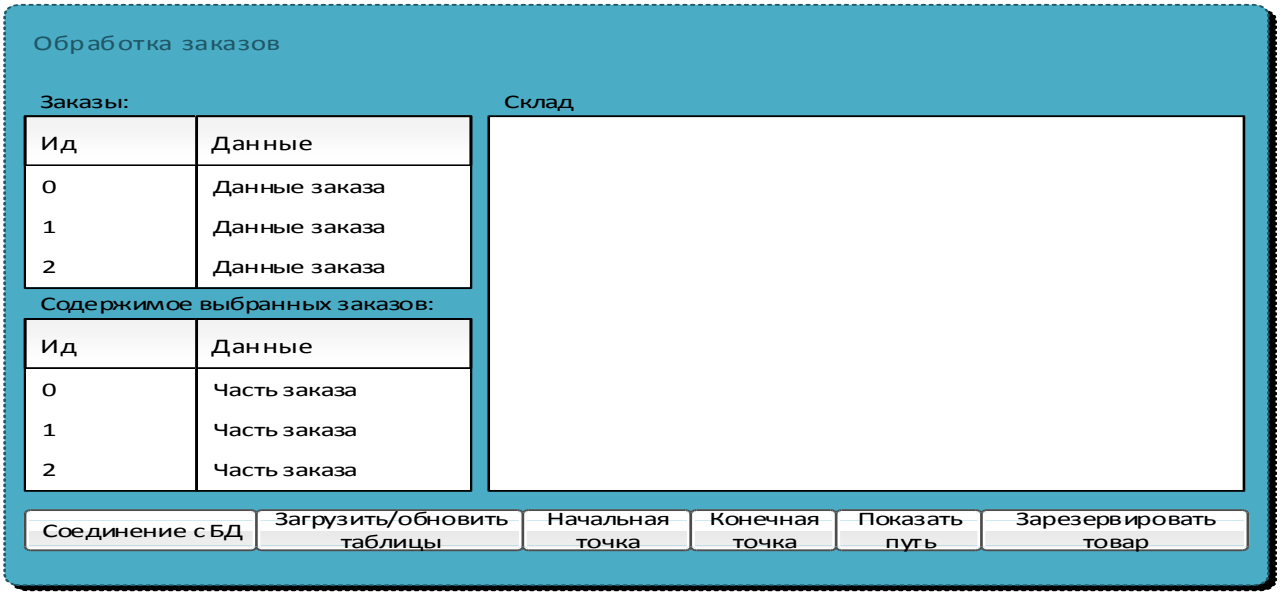


Рис 2.13. Структура меню обработки заказов.

Также должны быть окно выбора заказов, в котором можно было бы выделить необходимые заказы, и второе окно, содержащее состав выбранных заказов. Также необходимо третье окно, в котором схематично будут отображаться заказ и маршрут комплектации заказа. Исходя из сказанного



выше, была спроектирована следующая форма(см. рис. 2.14):

Рис 2.14. Форма обработки заказов.

3. ГЛАВА. РЕАЛИЗАЦИЯ СИСТЕМЫ КОМПЛЕКТАЦИИ ЗАКАЗОВ НА СКЛАДЕ

3.1. Реализация библиотеки классов нахождения кратчайшего пути

В этой библиотеке реализованы классы (их код вы можете увидеть в приложении):

1)Search_stock – поиск кратчайшего маршрута в графе через заданные точки с учетом того, что начальная и конечные точки не могут изменить свой порядок;

2)Database_stock – класс, реализующий взаимодействие с базой данных;

3)Shape – класс хранения объектов склада в памяти программы;

*4)*Way – класс хранения путей на складе в памяти программы;

Класс Search_stock

В этом классе реализуется поиск кратчайшего пути для конкретного маршрута и хранение графа. Содержит 5 приватных полей и 4 функции. Имеет следующие поля:

●int amount – количество вершин;

●bool flag_load – флаг о том, загружен ли граф;

●bool flag_search – найдены ли все кратчайшие пути;

●double[][] graph – матрица весов графа;

●int [][] way_graph – матрица хранения путей в графе.

Далее перечислены основные функции класса.

Функция: void load_graph(double[] graph, int count)

●назначение: занести в экземпляр класса матрицу весов;

●входные параметры

- double [] graph – матрица весов, записанная в одномерном массиве;

- count — количество вершин в графе;

Функция: int opt()

●назначение: нахождение кратчайших путей между вершинами алгоритмом Флойда с сохранением путей;

●значения: 0 успех, 1 граф не загружен.

Функция: int[] search(int amount_ft, int[] number, int in_ver, int fin_ver)

●назначение: поиск кратчайшего маршрута через указанные ключевые точки;

●входные параметры:

- int amount_ft — количество ключевых точек, кроме начальной и конечной;

- int[] number – массив ключевых вершин;

- int in_ver — начальная вершина;

- int fin_ver – конечная вершина;

●дополнительная информация: алгоритм описан в главе 2.

Класс Database_stock

В данном классе реализованы подключение к БД и выборка базы данных из БД. Этот класс для своей работы имеет набор полей, из которых 8 приватных и 5 публичных. При необходимости количество полей может быть расширено. Непосредственно перед работой с

к ней нужно подключиться. Класс, отвечающий за работу с БД, инициализируется строкой «подключение к базе», а также логином и паролем от базы данных. Поля следующие:

1.Приватные поля:

1)FileInfo file – содержит информацию о файле, о пути к нему для дальнейшей работы с ним;

2)int[] regtangle – содержит информацию о формате хранения в БД стеллажа;

3)int[] entrence – содержит информацию о формате хранения в БД

точки входа/выхода;

4)int[] stock – содержит информацию о формате хранения размеров склада;

5)int[] id_tags – содержит информацию о том, к какому ид текущие форматы хранения стеллажа, «точки входа/выхода»;

6)SqlDataAdapter masterDataAdapter — хранит запросы заполнения, обновления, вставки и удаления мастера таблицы;

7)SqlDataAdapter detailsDataAdapter — хранит аналогичные запросы для подчиненной таблицы, которая связана с мастер-таблицей через SqlDataAdapter;

8)SqlDataAdapter detailsDataAdapter2 — также хранит запросы для второй подчиненной таблицы, которая связана с первой подчиненной таблицей через аналогичный SqlDataAdapter;

9)string[] points массив точек которые необходимо посетить получается при показании пути.

2.Публичные поля:

1)Shapes shapes — класс лист фигур;

2)int[] size – содержит размеры склада ширину, высоту;

3)string[][] answer – хранит информацию по статусам заказа, необходима для работы корректного переключения статусов заказа с одного на другой;

4)Lines_Ways ways — класс лист путей;

5)SqlConnection – класс, в котором хранится соединение с БД.

Функция: databese_stock (string server, string database, string user, string password)

●назначение: инициализация класса database_stock;

●входные параметры:

- server — строка подключения к серверу;

- database — имя базы данных;

- user – имя пользователя;

- password – пароль этого пользователя в БД;

●Значение: экземпляр класса database_stock.

●Дополнительная информация: при инициализации класса специализируются такие поля, как shapes, ways, connection, size, regtangle, wall, entrence, id_tags.

Далее идут основные функции класса.

Функция: void GetData(BindingSource masterBindingSource, BindingSource detailsBindingSource, DataGridView detailtable)

●назначение: получить в masterBindingSourse данные мастер-таблицы, а в detailsBindingSource - данные подчиненной таблицы. Используется при обработки заказов;

●входные параметры:

- masterBindingSource — контейнер под хранение данных мастер таблицы (заказы);
- detailsBindingSource — контейнер под хранение данных подчиненной таблицы(содержимое заказов);
- detailtable — компонент для отображения подчиненной таблицы, необходим, чтобы убрать соединительную колонку;

●выходные параметры: masterBinndingSourse, DeatailsBingSourse, detailtable;

●дополнительная информация: запросы, названия таблиц и по каким колонкам их соединять лежит в настоечном файле для быстрой перенастройки без изменения кода.

Функция: DataSet GetData(DataSet data, BindingSource master, BindingSource details, BindingSource details2, DataGridView masterv, DataGridView detailv, DataGridView details2v, string name)

●назначение: получение 3 таблиц, подчиненных друг другу последовательно, с возможностью определения выбора, какие столбцы можно

редактировать, скрыть;

●входные параметры:

- DataSet data — хранит текущие значения таблиц, если они есть, необходим для изменения данных, в случае необходимости обновить данные в таблицах запускается с параметром null;

- BindingSource master, detail, details2 – контейнеры для хранения данных в трех таблицах;

- DataGridView masterv, detailv, details2v – компоненты отображения таблиц, необходимы в функции, чтобы скрыть и запретить изменения части столбцов;

●выходные параметры: master, detail, details2, masterv, detailsv, details2v;

●значения: dataset из трех таблиц, null в случае ошибки;

●дополнительная информация: аналогично верхней с разницей в том, что можно выбрать, какие столбцы скрыть, а какие запретить изменять, использует вспомогательную функцию shelf_return для получения массива команд с параметрами.

Функция: bool get_status()

●назначение: получить в поле класса answer возможные переходы с этого статуса на другие статусы их ид и название статуса, на который возможен переход;

●значения: true в случае успеха, иначе false;

●дополнительная информация в поле answer данные хранятся в следующем формате массив из 4 элементов {[Ид_текущего статуса];[Ид_статуса_перехода:Название_статуса _перехода (и дальше перечисление через пробел)][(Аналогично предыдущему)][(Аналогично предыдущему)]. В первой ячейке массива хранится ид текущего статуса, с которого возможен переход. Во второй ячейке ид статусов, на которые возможно перейти, повышая статус заказа. В третьей ячейке хранятся данные,

аналогичные второй ячейке, только понижая статус заказа. А в последней, четвертой ячейке, переходы на статусы ошибки. Использует для получения названия статуса функцию returnname с параметрами (Ид_статуса).

Функция: int swap_status (Int16 id, Int16 id2)

●назначение: перевести заказ с одного статуса на другой;

●входные значения:

- int16 id — ид заказа, статус которого можем перевести.

- Int16 id2 — ид статуса, на который хотим перевести.

●значения: 0 успех, -1 нет нужной части настоечного файла, 1 sql ошибка.

Функция: int load_stock()

●назначение: произвести загрузку объектов склада из БД в поле Stock;

●значение: 0 успех, -1 если нет нужной части настоечного файла, -2 нет файла, 1 sql ошибка в запросе, 2 ошибка при преобразовании значений;

●дополнительная информация: использует функцию parse класса с параметрами (полное_имя_объекта, дополнительная_информация).

Функция int load_Ways()

●назначение: произвести загрузку путей из БД в поле ways;

●значение: 0 успех, -1 если нет нужной части настоечного файла, -2 нет файла, 1, 2 sql ошибки в первом или втором запросе соответственно.

Функция int save_in_database_stock (Shapes vshapes)

●назначение: обновление информации об объектах склада в БД, сохранение новых данных;

●входные параметры: vshapes — лист объектов склада, который будет сохранен;

●значения: 0 успех, -2 нет файла, -1 не нужной части файла, 1-4 sql ошибка, связанная с соответственным запросом;

●дополнительная информация: использует функцию parse класса с

параметрами (полное_имя_объекта, дополнительная_информация) и функцию класса parse с параметрами (объект_склада), для меньшего количества перезаписей использует алгоритм с анализом строк для имзенения.

Функция: int save_in_database_ways (Lines_Ways vways)

●назначение: обновление информации о путях на складе в БД, сохранение новых данных;

●входные данные: vways лист текущих путей на складе;

●значения: 0 успех, -2 нет файла, -1 нет нужной части настоечного файла, в остальных случаях sql ошибка, номер запроса которой будет возвращаемым значением;

●дополнительная информация: работает по похожему алгоритму на алгоритм сохранения объектов склада.

Функция: int build_shortcut (int []id, string begin, string end)

●назначение: выдать и запомнить кратчайший маршрут комплетации заказа для следующей возможной комплектации и резервирования продукции;

●входные параметры:

- id — массив ид заказов, которые нужно собрать;
- begin и end точки начала/конца сбора заказа;

●значения: 0 успех, -2 нет файла, -1 нет нужного участка настоечного файла 1-4 sql ошибки по номерам запросов;

●дополнительная информация: использует алгоритм нахождения кратчайшего маршрута, описанный в главе 2. Отличается учётом возможности того, что товар под заказ уже зарезервирован, а значит, к путям нужно просто добавить те точки, которые необходимы в этом заказе.

Класс shape

Этот класс имеет несколько наследников для более простой работы с ним. Сам класс представлен в виде10 полей и 4 функций. Имеет следующие поля:

1.Приватное поле:

1)string full_name_reserve — полное имя без номера.

2.Публичные поля:

1)object addinfo — поля для прикрепления дополнительной информации;

2)string Name – название типа объекта склада;

3)Signature sign – поле, которое используется в объекте типа «шкаф» для связи с объектом, указывающим на место доступа к нему;

4)int mode – показывает сторону доступа к стеллажу или направление точки входа;

5)int Num_shelf – номер объекта склада;

6)Pen Pen – то, чем будет нарисовано обрамление фигуры;

7)Brush Brush – заполнитель фигуры;

8)Rectangle Rectangle – для хранения данных прямоугольника;

9)GraphicsPath GraphicsPath – хранит графическое отображение.

Класс Ways

Данный класс хранит данные о путях. Он имеет 8 публичных полей полей и 3 функции, включая конструктор. Его поля следующие:

1)bool hidden — флаг, который показывает, скрывать ли путь;

2)int Num – количество точек в этом пути;

3)double sum – вес пути;

4)string[] way – хранит имена точек начала и конца пути;

5)Pen Pen – ручка, которой будет нарисован путь;

6)List<Point> Points – лист точек на этом пути;

7)ToolStipMenuItem MenuItem – ссылка на menuitem, который работает с данным путем;

8)GraphicsPath GraphicsPath – хранит графическое отображение.

Конструктор: public Way(int x, int y)

●назначение: инициализация класса с первой введенной точкой

●входные параметры: координаты x,y первой точки.

Функция: void add_line(Point line)

●добавить ещё линию в путь, состоящий из ломаных линий;

●входной параметр: координаты следующей точки.

3.2. Реализация ПО подготовки данных.

Исходя из описанного в главе 2.5, становится понятно, что должно быть на форме ПО подготовки данных, в том числе и его структура. Следовательно, можно приступать к его реализации. В итоге была получена следующая форма(см. рис. 3.1):

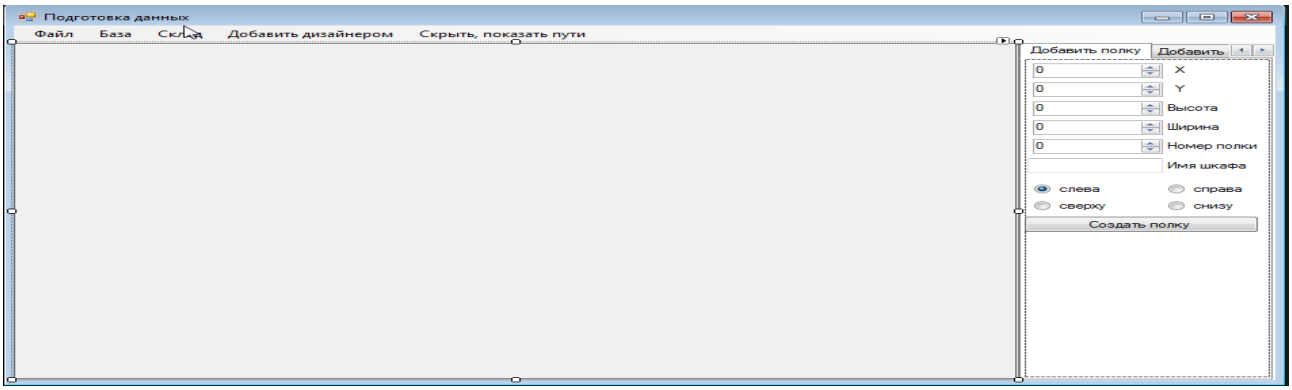


Рис. 3.1. Форма подготовки данных

В меню файла из предыдущей главы понятно, что там должно быть два пункта меню. Кнопка «сохранить» приводит к тому, что появляется диалоговое окно сохранения, которое предлагает выбрать файл для сохранения. Аналогично кнопка загрузки приводит к появлению диалогового окна загрузки. Их обработчики соответственно: void загрузитьСкладToolStripMenuItem

_Click(object sender, EventArgs e), void сохранитьСкладToolStripMenuItem _Click(object sender, EventArgs e).

Следующая вкладка БД. Прежде чем загружать с БД данные, необходимо сначала к ней подключиться. При нажатии на кнопку соединения с БД появится следующая форма(см. рис. 3.2):

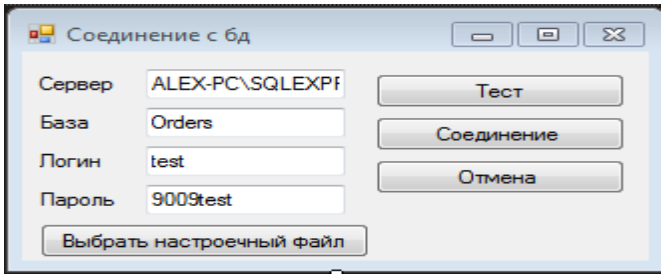


Рис. 3.2. Форма параметров соединения с БД

При нажатии на этой форме на кнопку «тест» будет сделана проверка успешного подключения. В этом случае программа выдаст сообщение, что тест пройден или то, что параметры неверны. Отмена или крестик просто закрывают форму, поэтому ничего не происходит. Кнопка «выбрать настоечный файл» открывает меню открытия файла для выбора настоечного файла (файла с запросами к БД). Кнопка «соединение» отправит текущие параметры в предыдущую форму. Это инициализирует класс, отвечающий за соединение, работу с БД. За это отвечает обработчик void соединениеСБазойToolStripMenuItem_Click(object sender, EventArgs e). За тестирование соединения отвечает обработчик void button1_Click(object sender, EventArgs e). За выбор настроечного файла отвечает обработчик void button4_Click(object sender, EventArgs e). За кнопку соединения отвечает обработчик void button2_Click(object sender, EventArgs e). За кнопку отмена отвечает обработчик void button3_Click(object sender, EventArgs e).

Далее идет пункт меню «склад», в нем кнопка «новый склад». Она очищает «текущий склад» до пустого, а затем меняет его размеры на указанные

на вызванной форме (см. рис. 3.3).

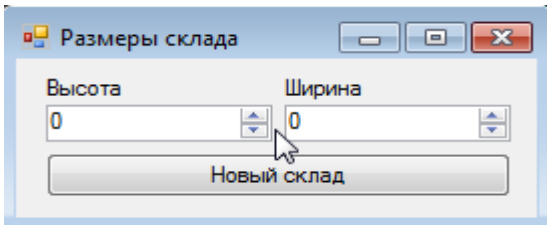


Рис. 3.3. Параметры размеров нового склада.

За это отвечает обработчик void создатьСкладToolStripMenuItem _Click(object sender, EventArgs e). Следующая кнопка «очистить склад» делает тоже самое, что и предыдущая, но не изменяет размеров и не вызывает форму. Это делает обработчик void очиститьToolStripMenuItem1_Click(object sender, EventArgs e).

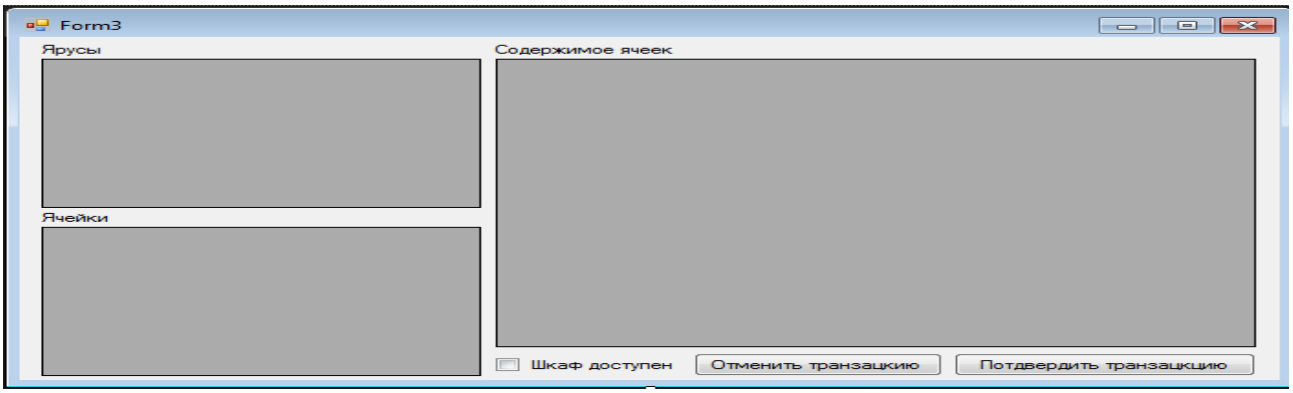
Затем следует кнопка «начало меню скрыть/показать пути», которая меняет флаг, отвечающий за отображение путей. Она является меню и кнопкой одновременно. Как кнопка она является переключателем отображения путей, и её обработчик – это void скрытьПоказатьПутиToolStripMenuItem_Click(object sender, EventArgs e). У этой кнопки в случае наличия путей появляется подменю с количеством пунктов, равному количеству путей, в котором каждый пункт отвечает за то, чтобы спрятать или показать конкретный путь. У этого пункта, в свою очередь, есть подпункт «удалить путь», который, соответственно, удаляет этот путь. У созданных подпунктов обработка кнопок производится через следующие обработчики: void create_menustip() создание пункта меню и его подменю, void hidden_line_way(object sender, EventArgs e) скрыть показать конкретный путь, void Clear_way(object sender, EventArgs e) очистка пункта меню. Добавление, запоминание пути производится в функции клика мыши в обработчике void AreaDrawing_MouseClick(object sender, MouseEventArgs e).

Далее следует дополнительное меню с 3 вкладками, описанными в проектировании. У каждой вкладки есть по кнопке, которая отвечает за добавление по координатам объекта вкладки. Они все похожи друг на друга, но если добавляется шкаф, то создается два объекта, связанных друг с другом: это

сам шкаф и место доступа (сторона, с которой можно брать товар) к шкафу. Их обработчики: void button_shelf_Click(object sender, EventArgs e), void button2_wall_click(object sender, EventArgs e), void create_entrens_Click(object sender, EventArgs e).

Далее следует функция «клик мыши по области склада», немного описанная ранее. В зависимости от кнопки, которой это было сделано, будет или удаление объекта склада и связанных с этим объектом путей в случае клика правой кнопкой мыши по ним (а если объектов под курсором не было, то ничего не будет сделано). Или же, в случае левой кнопки мыши, при нажатии на шкаф будет вызвана форма просмотра шкафа. В случае же нажатия на точку входа/выхода или место доступа к шкафу будет начато построение пути, которое будет закончено, когда будет нажатие на объекты такого же типа. Пока идет построение пути, клик левой кнопкой мыши будет строить следующий участок пути. В случае же, если дизайнером было выбрано добавление, то при клике левой кнопкой мыши будет начато и закончено рисование нового объекта склада, а правая кнопка мыши будет работать без изменений. За это отвечает обработчик void AreaDrawing_MouseClick(object sender, MouseEventArgs e).

Форма, упомянутая выше, выглядит, как указано на рисунке 3.4. Она позволяет просматривать, добавлять и изменять параметры шкафа, в том числе



и доступность самого шкафа.

Рис. 3.4. Форма просмотра и изменения содержимого шкафа.

Она имеет функции обновления этих таблиц и их изменения. Обработчик отмены изменений cancel(object sender, EventArgs e). Обработчик подтверждения void accept(object sender, EventArgs e).

3.3. Реализация ПО комплектации заказов.

Исходя из описанного в главе 2.5 становится понятно, какой должна быть форма ПО обработки заказов, и потому можно приступать к её реализации. В итоге была реализована следующая форма(см. рис. 3.5):

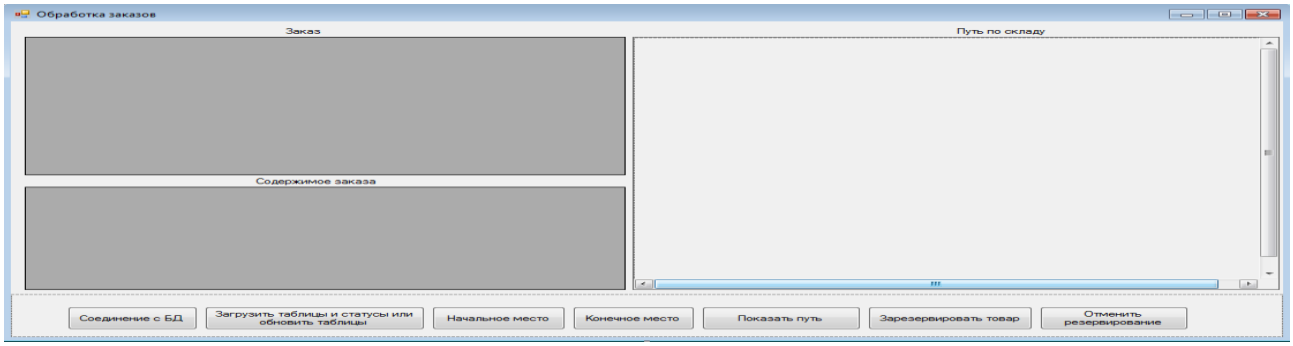


Рис. 3.5. Форма комплектации заказов.

Первая кнопка «соединение с БД» даст нам тот же результат, что и аналогичная кнопка в ПО подготовки данных(см. Рис. 3.2). Отличие будет лишь в том, что будет загружена информация, с какого на какой статус может быть переведен заказ. Обработчик кнопки void entrence_Click(object sender, EventArgs e). А за изменение возможных переходов отвечает обработчик void mastertable_SelectionChanged(object sender, EventArgs e).

Кнопка «загрузить таблицы и статусы» обращается через специальный класс к Базе данных с целью получить заказы и их содержимое и устанавливает взаимосвязь заказа и его содержимого. Поэтому в нижней таблице будет отображаться содержимое последнего выбранного заказа. В случае повторного нажатия обновляет таблицы. Обработчик void button2_Click(object sender, EventArgs e).

Кнопки «начальная точка» и «конечная точка» обозначают, где должен

начинаться и заканчиваться маршрут комплетовщика. Их обработчики соответственно void begin_Click(object sender, EventArgs e), void button1_Click(object sender, EventArgs e)

Кнопка «показать путь» получает массив ид выбранных заказов и запускает функцию постройки кратчайшего маршрута. Если необходимо, после этого можно зарезервировать выбранные товары для того, чтобы программы не пыталась отобрать их снова. В случае, если товары для заказа уже зарезервированы, необходимые стеллажи просто будут добавлены к необходимым точкам посещения. В случае изменения выбранных заказов при нажатии на эту кнопку будет повторно запущен поиск кратчайшего пути, и только после этого повторное нажатие на эту кнопку зарезервирует товар. Обработчик этой кнопки void build_Click(object sender, EventArgs e).

На определенных статусах можно отменить резервирование товара, т.к. по статусу предполагается, что он ещё не собран. В таком случае существует возможность, что заказ может быть собран более коротким образом. За смену статусов отвечает обработчик void swap_status(object sender, EventArgs e). За резервирование отвечает reservid_Click(object sender, EventArgs e). За отмену резервирования отвечает функция cansel_reservid(object sender, EventArgs e)

Полный код всех функций можно увидеть в приложении.

3.4. Тестирование

Для основной задачи необходимо подготовить данные, поэтому сначала будет протестировано ПО подготовки данных и только потом ПО обработки заказов.

В ПО подготовке заказов необходимые функции указаны в главе 2.5. Поэтому\ пойдем в их тестировании по порядку. Для начала сделаем несколько объектов склада и путь между ними, после чего сохраним. А затем проверим, загрузятся ли данные. Но для этого сначала нужно проверить добавление путей

и объектов склада. Итог добавления виден на рисунке 3.6.

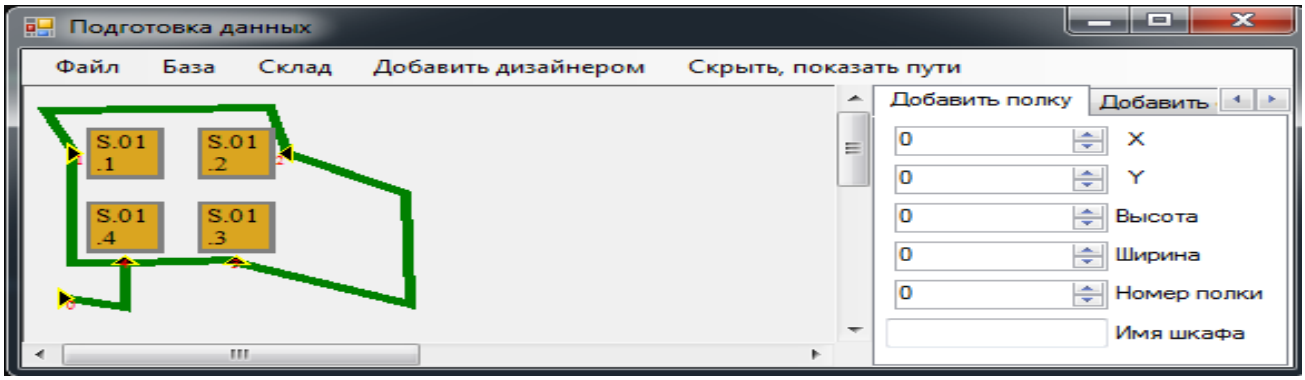


Рис. 3.6. Проверка добавления объектов склада и путей.

Было протестировано добавление как в обычном режиме, так и дизайнером. Следующим действием будет произведено сохранение склада (см. рис. 3.7, 3.8).

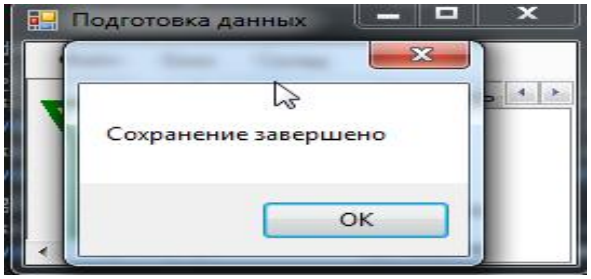


Рис. 3.7. Окончание сохранения файла

После этого проверим, будет ли успешной загрузка сохраненного файла(см. рис. 3.8).

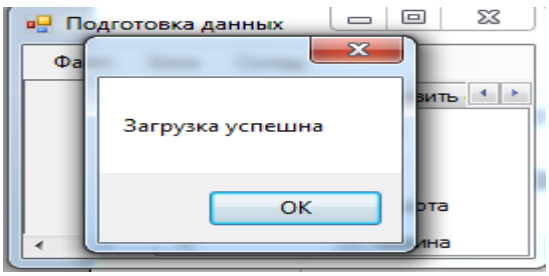


Рис 3.8. Окончание загрузки файла

После того, как провели загрузку из файла и убедились в том, что она прошла успешно, будем тестировать соединение с БД. Нажмем на кнопку «база» и увидим, что перед нами окно ввода данных о подключении к БД.

Форму подключения можно увидеть ниже, на рисунке 3.9

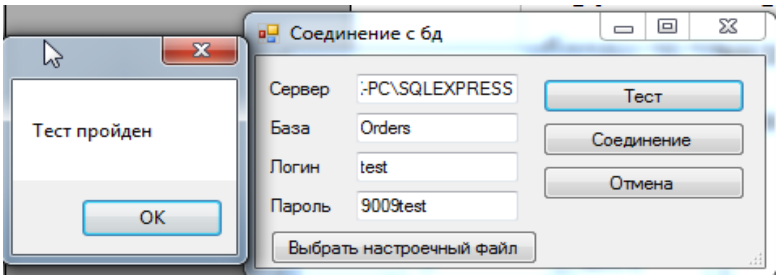


Рис 3.9. Форма подключения к БД

Теперь стали доступны кнопки выгрузки склада из БД и его загрузки. Произведем сначала загрузку БД из файла, а потом сохраним склад. Итог загрузки из БД показан на рисунке ниже(см. рис. 3.10).

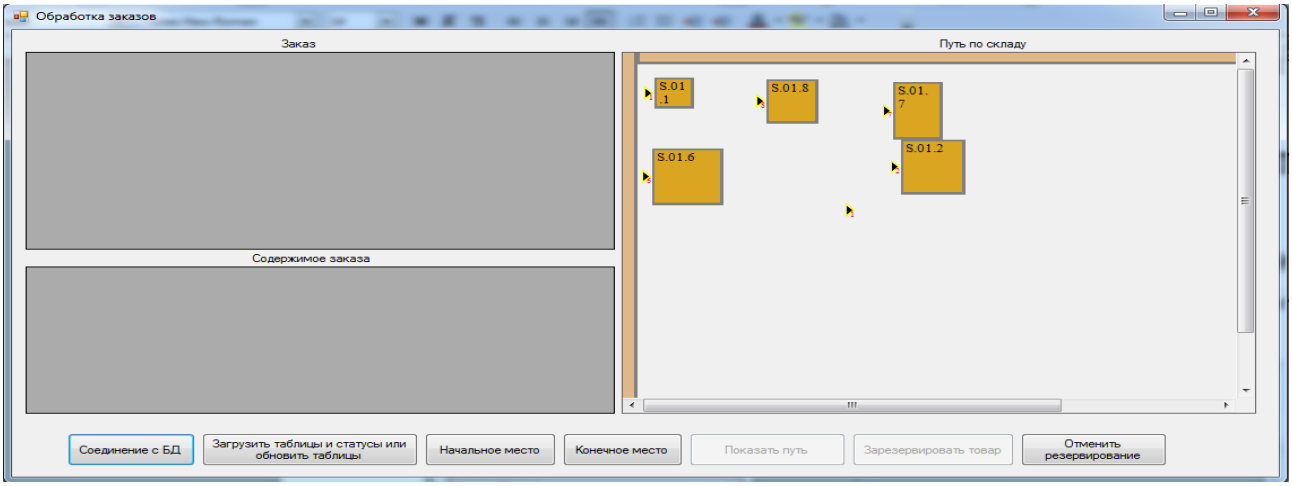


Рис 3.10. Загрузка склада из БД

После этого производим сохранение склада. Так как различий больших нет, снимок экрана пропускаем. Далее проверяем, производится ли просмотр данных шкафа при нажатии по нему. И при нажатии на стеллаж появляется форма, представленная на рисунке 3.11.

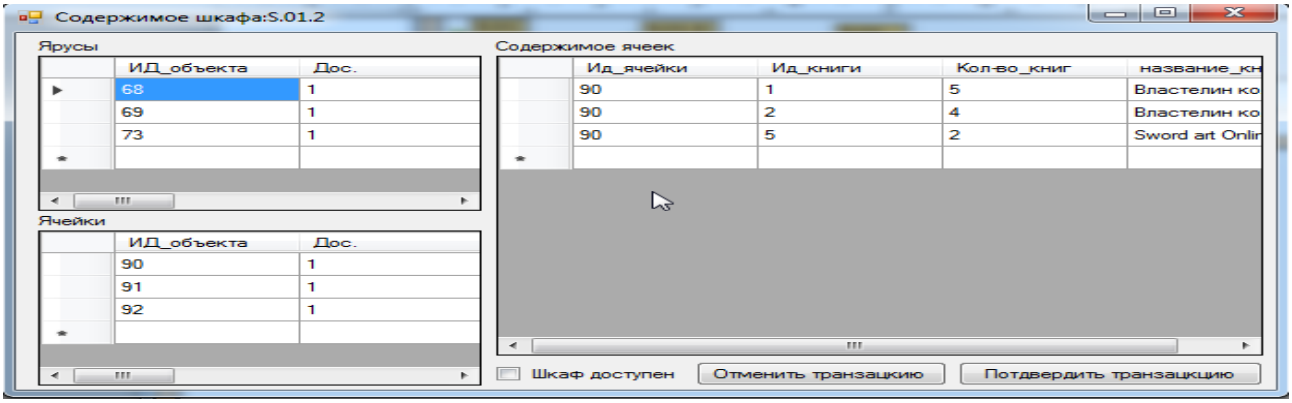


Рис 3.11. содержимое S.01.2

На этой форме мы видим ярусы стеллажа, их ячейки и содержимое этих ячеек. Также была проверена возможность добавления удаления строк, их содержимого и связанных с ними данных.

Следующим будет протестировано ПО обработки заказов. При запуске приложения появляется форма (см. рис. 3.12):

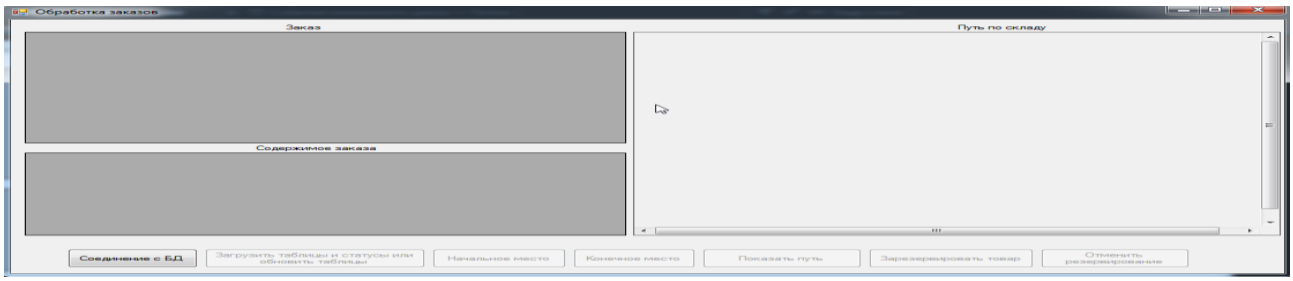


Рис 3.12. Запущенная форма обработки заказов.

Поскольку ещё не произведено подключение к БД, остальные кнопки заблокированы. Нажимаю кнопку «соединение с БД». Появится форма, аналогичная ранее описанной, выглядящая также, как на рисунке 3.10. с аналогичным функционалом и функциями. После этого разблокируется кнопка «загрузить таблицы и статусы или обновить таблицы». Итог нажатия и успешного соединения виден на рисунке 3.13.

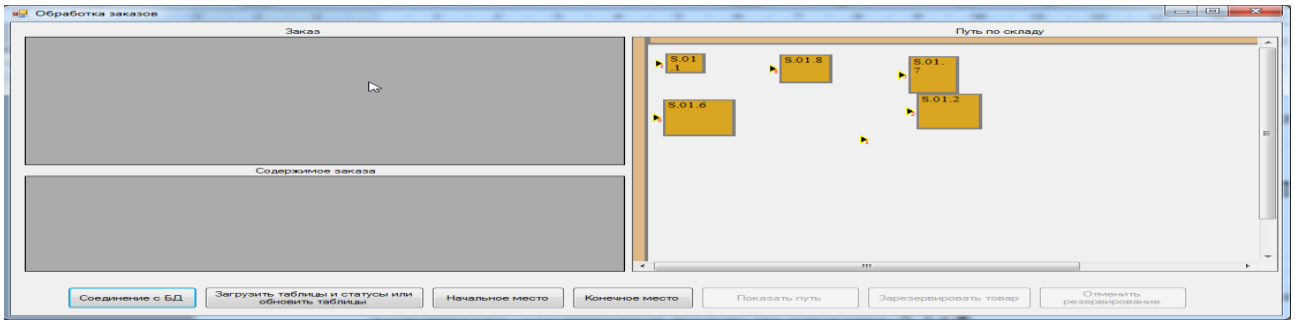


Рис 3.13. форма после соединения.

После чего произведем загрузку таблицы, нажав вторую кнопку слева. Это приведет к заполнению таблицы, и итог будет виден на рисунке 3.14.

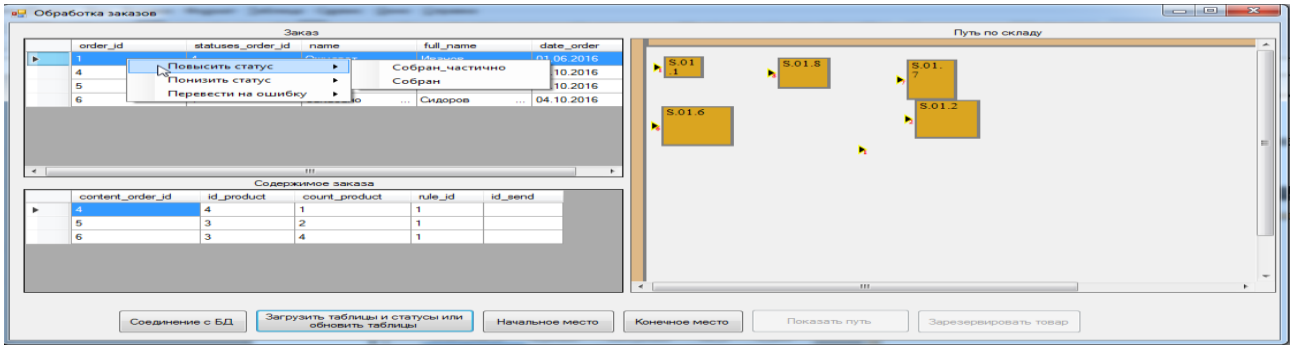
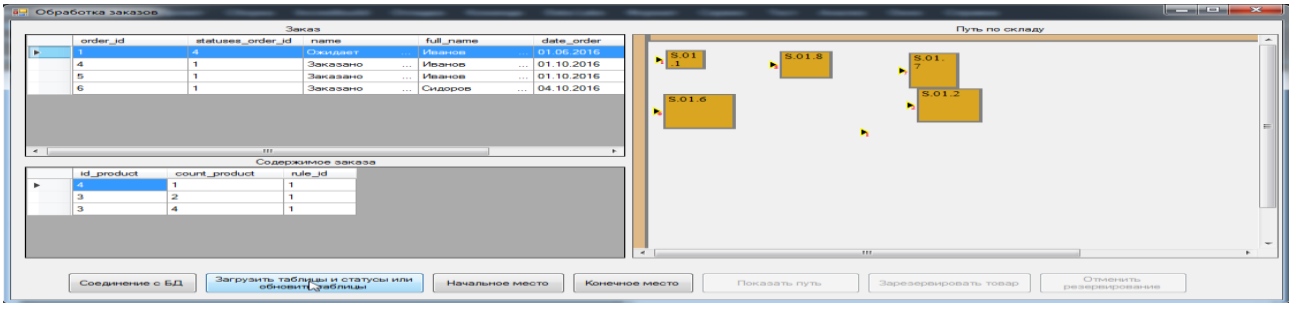


Рис 3.14. Форма с загруженными таблицами

В итоге перед нами содержимое выбранного заказа. И видно то, что мы можем перевести его с одного статуса на другой. Перед тем, как разблокируется кнопка «показать путь», необходимо выбрать начальное и конечное место. После нажатия и выбора начальной и конечной точки мы увидим то, что кнопка



«показать путь» разблокировалась (см. рис. 3.15).

Рис. 3.15. форма после выбора начальной и конечной точки

После того, как мы нажмем «показать путь», на области склада зеленой линией отобразится кратчайший маршрут для комплектации заказа, а затем нажмём кнопку зарезервировать заказ. В итоге мы получим то что товар зарезервирован и для сбора получен кратчайший маршрут. Что мы и видим на рисунке 3.16.

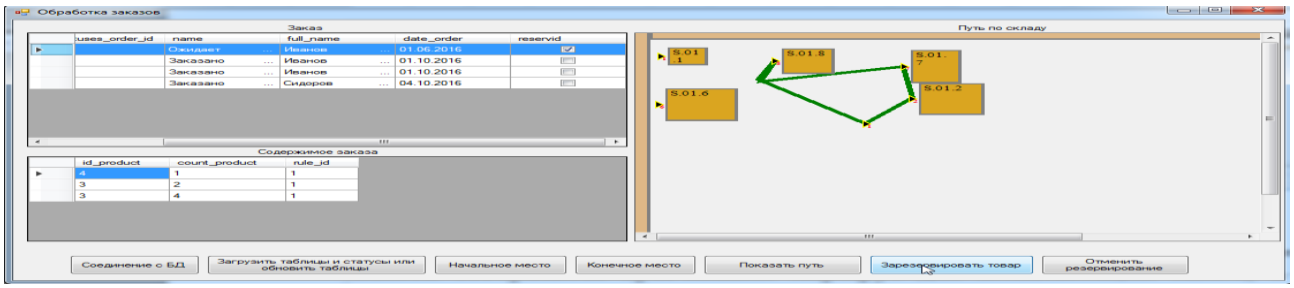
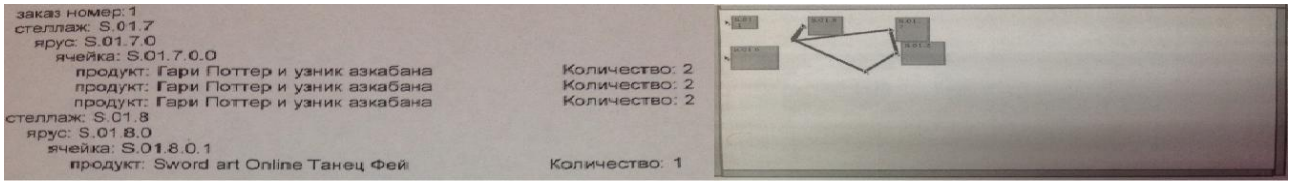


Рис. 3.16. Показ пути и резервирования товара на форме

Также при резервировании пути производится предложение напечать путь



а также с каких полок что собрать, итог можно увидеть на рисунке 3.17

Рис 3.17. путь и комплектация товара

ЗАКЛЮЧЕНИЕ

Сейчас существует проблема в автоматизации складов у мелких и средних предприятий, особенно там, где не требуется высокая степень автоматизации или нет возможности ввести много дополнительного оборудования. Единственное исключение – это 1С, но у него закрытый код, а то, что можно добавлять и изменять, пишется на языке 1С. Поэтому необходимость разработки нового программного обеспечения для низкой автоматизации складов, которое было бы просто настроить и в случае необходимости изменить, а также было бы общедоступным, очевидна.

В данном отчете преддипломной практики была поставлена цель проанализировать существующее ПО комплектации заказа, а также разработать собственный алгоритм комплектации заказа, который позволит производить её быстрее.

В ходе разработки программы были решены следующие задачи:

1) разработан алгоритм комплектации заказов на складе;
2) был обоснован выбор программных средств;
3) разработана модель базы данных;
4) спроектирована структура приложения;
5) спроектирован интерфейс пользователя;
6) реализован в виде библиотеки классов алгоритм нахождения кратчайшего маршрута;
7) реализовано ПО подготовки данных;
8) реализовано ПО обработки заказов;
9) произведена проверка работы системы на тестовых заданиях.

Таким образом, поставленная цель работы была достигнута.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1.Виды комплектации заказов [Электронный ресурс] //URL: https://snarta.com/author/de/ (Дата обращения 5.06.2017)

2.Процессы связанные с комплектацией заказа [Электронный ресурс] //URL: https://olegon.ru/showthread.php?t=23047 (Дата обращения 5.06.2017)

3.Сравнения систем WMS [Электронный ресурс] //URL: http://logist.ru/articles/vybor-sistemy-upravleniya-skladom-wms-v-2016-godu (Дата обращения 5.06.2017)

4. Официальный сайт Аваго[Электронный ресурс] //URL: http://www.avacco.ru/ (Дата обращения 5.06.2017)

5.Причины выбрать язык определенного уровня программирования [Электронный ресурс] //URL: https://tproger.ru/translations/programming-languages-types/ (Дата обращения 5.06.2017)

6.Википедия, алгоритм Флойда[Электронный ресурс] //URL: https://ru.wiki pedia.org/wiki/Алгоритм_Флойда_-_Уоршела (Дата обращения 5.06.2017)

7.Достоинства MS SQL server [Электронный ресурс] //URL: http://users.fre enet.am/~tank1873/Microsoft%20SQL%20Server/benefits.htm (Дата обращения 5.06.2017)

8.Параллельный цикл for в C# [Электронный ресурс] //URL: https://msdn.microsoft.com/ru-ru/library/system.threading .tasks.parallel(v=vs.110) .aspx (Дата обращения 5.06.2017

9.Гаджинский, А. М. Современный склад. Организация, Технологии, управление и логистика: учебно-практическое пособие / А. М. Гаджинский — М.: Проспект, 2005.- 176 с.

10.Волгин, В. В. Склад. Организация. Управление. Логистика / В. В. Волгин. М.: Дашков и К0, 2006. - 732 с.