

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ И ДИЗАЙНА»

Кафедра информационных технологий

**Методические указания и задания
к контрольной работе № 3**
по дисциплине «**Алгоритмизация и программирование**»
направление подготовки 09.03.03 «Прикладная информатика»
профиль *Прикладная информатика в экономике*
очной формы обучения

Составитель:

О. В. Кофнов

Санкт-Петербург

2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
1. ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ.....	6
2. РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ ДЛЯ АВТОМАТИЗАЦИИ ХОЗЯЙСТВЕННОЙ ДЕЯТЕЛЬНОСТИ	6
3. ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА И ВСТРОЕННЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ	7
4. ОСНОВНЫЕ КОНСТРУКЦИИ ВСТРОЕННОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ 1С.....	9
4.1. Структура программных модулей, типы данных и базовые операторы	9
4.2. Общие и прикладные объекты, универсальные коллекции значений.....	14
4.3. Взаимодействие с пользователем, ввод и вывод данных.....	17
5. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ПЛАТФОРМЕ «1С:ПРЕДПРИЯТИЕ 8.3»	22
5.1. Разработка архитектуры прикладного решения, создание необходимых констант, справочников и документов	23
5.2. Создание регистров накопления.....	29
5.3. Создание печатных форм и отчетов.....	33
5.4. Создание ролей.....	41
6. КОНТРОЛЬНЫЕ ЗАДАНИЯ	43
6.1. Учет товаров в разрезе складов	43
6.2. Учет себестоимости в пределах компании	44
6.3. Учет себестоимости в разрезе складов	45
6.4. Контроль себестоимости остатков товаров.....	45
6.5. Расчет прибыли при списании по методу FIFO.....	46
6.6. Учет при списании сроков годности	46
6.7. Указание конкретных сроков годности при списании товаров	47
6.8. Резервирование товаров	48
6.9. Расчеты с покупателями	48
6.10. Дни рождения студентов	49
ЗАКЛЮЧЕНИЕ	51
ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ	52
СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ	53

ВВЕДЕНИЕ

Целью изучения дисциплины является формирование у студентов компетенций в области фундаментальных представлений об алгоритмизации разработки программных систем и современных методах промышленного программирования прикладных экономических, отраслевых и профильных задач. В процессе изучения курса студенты получают представление об основных понятиях теории алгоритмов и методах алгоритмизации для решения прикладных задач; знакомятся с основными формами алгоритмического представления и программной обработки данных в различных прикладных инфокоммуникационных системах и интегрированных средах промышленного назначения; овладевают современными высокоуровневыми программными средами и инструментами разработки для программного решения прикладных задач различной степени сложности; вырабатывают навыки углубленного стилевого программирования, квалиметрии и планирования процесса коллективной разработки, применения промышленных отраслевых стандартов для поддержки инжинирингового процесса в сложных информационных системах.

В процессе изучения дисциплины студент выполняет ряд практических заданий, а также две контрольные работы, посвященные развитию навыков алгоритмизации и программирования в различных средах разработки с использованием различных языков программирования высокого уровня.

В данном пособии рассматривается методический материал и задания ко второй контрольной работе, посвященной разработке предметно-ориентированных приложений на технологической платформе «1С:Предприятие 8» с использованием встроенного языка программирования.

1. ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

Контрольная работа, выполняемая в период между сессиями в установленные учебным графиком сроки, заключается в выполнении **одного комплексного** задания, включающего разработку конфигурации прикладного приложения на платформе «1С:Предприятие 8.3» и реализовать в ней необходимые алгоритмы автоматизации учета заданной хозяйственной деятельности. Отчет по выполнению контрольной работы должен содержать разделы:

- введение;
- описание особенностей заданной области автоматизации хозяйственной деятельности;
- описание создаваемой архитектуры метаданных в разрабатываемой конфигурации и обоснование этой архитектуры;
- описание и обоснование используемых алгоритмов на языке 1С для автоматизации учета, обоснование выбора программных модулей конфигурации для размещения кода этих алгоритмов;
- описание проверки работоспособности разработанной конфигурации и программных модулей для различных наборов учетных данных;
- краткое руководство пользователя для работы с созданной конфигурацией;
- ограничения на использование созданного программного приложения, допущения и упрощения, сделанные при его разработке;
- заключение, содержащее оценку эффективности созданного приложения, рекомендации по его использованию хозяйствующими субъектами.

Все разделы и этапы работы следует сопровождать копиями экрана с соответствующими пояснениями и интерпретацией результатов.

2. РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ ДЛЯ АВТОМАТИЗАЦИИ ХОЗЯЙСТВЕННОЙ ДЕЯТЕЛЬНОСТИ

При бурном развитии современной экономики, огромном потоке информации, сопровождающем любую хозяйственную деятельность, актуальным является вопрос о своевременной обработке этой информации и принятии на её основе верных управленческих решений. Такая обработка должна происходить с минимальными экономическими затратами для предприятия. Поэтому с переходом к рыночным отношениям в нашей стране востребованными стали различные компьютерные системы, автоматизирующие учет хозяйственной деятельности в самых различных областях.

Видов деятельности ограниченное количество, и многие организации и частные предприниматели, казалось бы, осуществляют абсолютно одинаковые последовательности хозяйственных операций. Однако использование универсальных программных продуктов для автоматизации учета такой деятельности не встретило одобрения со стороны конечных пользователей из-за существующих нюансов в работе каждого хозяйствующего субъекта, что подчас является его конкурентным преимуществом. В то же время разработка индивидуальных

программных продуктов и комплексов является делом долгим, сложным, затратным и далеко не всегда дающим нужный результат. Малые предприятия и частные предприниматели не могут пойти на такие траты. Кроме того, для предприятий любого уровня при использовании продукта, сделанного «под заказ», встает проблема дальнейшего сопровождения и развития системы, в том числе и на технологическом уровне, что связано с развитием компьютеров, появлением новых операционных систем и так далее. Поэтому оптимальным, особенно для Российской Федерации, явилось появление программных продуктов и технологических платформ для автоматизации ведения хозяйственного учета и управления предприятием с возможностью глубокой тонкой настройки и доработки под нужды заказчика, например, «1С:Предприятие 8». Данная технологическая платформа с одной стороны предоставляет богатый набор инструментов и заготовок, называемых метаданными, для конструирования собственной учетной системы, с другой стороны, позволяет реализовывать алгоритмы любой сложности на встроенном языке 1С.

3. ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА И ВСТРОЕННЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ

Прикладное приложение «1С:Предприятие 8» состоит из двух основных компонент: технологической программной платформы и информационной базы, содержащей исполняемую конфигурацию. По аналогии с Java можно сказать, что технологическая платформа представляет собой виртуальную машину, на которой исполняется скомпилированный байт-код программных модулей конфигурации. Некоторой особенностью является лишь то, что конфигурация реализуется в составе информационной базы 1С, представляющей собой одновременно и базу данных, и хранилище исполняемого программного кода. Конфигурация может быть выгружена из одной информационной базы и загружена в другую (с одновременным изменением её структуры в соответствии с загружаемой конфигурацией), но исполняться отдельно от информационной базы не может. Исключение составляют внешние отчеты и обработки, которые сохраняются в виде отдельных файлов и могут быть запущены в информационных базах с различающимися конфигурациями.

При создании новой информационной базы без использования шаблона какой-либо типовой конфигурации её конфигурация пустая. Она содержит лишь ветки объектов метаданных: Общие, Константы, Справочники, Документы и так далее. 1С строго говоря не является объектно-ориентированной системой, однако с помощью интерактивного конструирования в конфигураторе можно создавать свои объекты метаданных видов, соответствующих веткам конфигурации, что аналогично созданию новых классов на основании родительских классов при объектно-ориентированном программировании. Программные методы и свойства, имеющиеся у родительской ветки, доступны и у располагающегося в ней объекта метаданных, для которого разработчик может определить новые свойства (реквизиты, табличные части) и программные ме-

тоды, располагающиеся в соответствующих модулях созданного объекта метаданных. По аналогии с объектно-ориентированным программированием у видов объектов метаданных существуют свои предопределенные обработчики событий, код которых разработчик для своего объекта метаданных может написать самостоятельно, изменив таким образом функциональность системы.

Информационную базу с простейшей конфигурацией для занесения, хранения и просмотра данных можно создать без строчки программного кода, используя лишь интерактивный конструктор метаданных в составе конфигуратора. Однако для автоматизации обработки данных, упрощения работы пользователя с системой, автоматического формирования аналитических отчетов объекты метаданных имеют программные модули, в которых в виде отдельных процедур и функций помещается созданный разработчиком код на встроенном языке программирования 1С. Часть этих процедур имеют предопределенные имена, что соответствует обработчикам событий, вызываемых технологической платформой. В свою очередь эти процедуры вызывают процедуры и функции, описанные разработчиком, и так далее.

Так как «1С:Предприятие» разработана в России, то и в синтаксисе встроенного языка используются буквы русского алфавита и русские слова. Для разработчиков на иных языках программирования это делает язык 1С необычным и затрудняет его освоение. Однако платформа «1С:Предприятие 8» позволяет вести разработку и на английском языке: все операторы языка и предопределенные свойства и методы видов объектов метаданных имеют англоязычные синонимы, в именах переменных можно использовать латиницу и кириллицу, при этом никаких переключений в настройках конфигуратора делать не надо – просто пишете на удобном для вас языке, даже можете смешивать русские и английские слова. Однако учтите, что в силу исторических особенностей большая часть кода для 1С написана «по-русски», а смешивание латиницы и кириллицы замедляет разработку и затрудняет читаемость кода.

Для удобства разработчика в конфигуратор встроен «Синтаксис-помощник» (рис. 3.1).

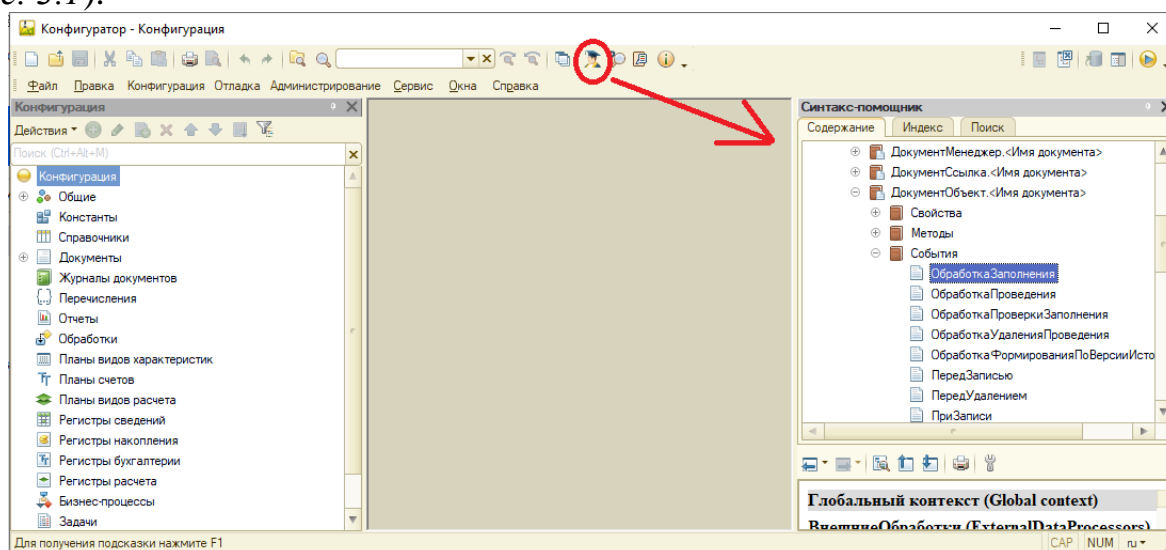


Рис. 3.1. Окно информационной базы с пустой конфигурацией и «Синтаксис-помощник» в режиме конфигуратора

В «Синтакс-помощнике» содержится полное описание встроенного языка программирования 1С с примерами и привязкой к универсальным и прикладным объектам конфигурации. В конфигураторе реализован поиск и контекстный поиск в «Синтакс-помощнике».

Хотя конфигуратор 1С и является по-прежнему основным инструментом для разработки и модернизирования конфигураций, в настоящее время существуют сторонние инструменты разработки, в частности, «1С:EDT», по интерфейсу напоминающие популярные интегрированные среды разработки на иных языках программирования. Подробнее о возможностях и особенностях «1С:EDT» можно узнать на сайте компании «1С».

4. ОСНОВНЫЕ КОНСТРУКЦИИ ВСТРОЕННОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ 1С

Язык программирования «1С:Предприятия 8» является языком программирования высокого уровня, компилируемым в байт-код, без строгой типизации (динамическая типизация), поддерживает все основные конструкции структурного программирования (ветви, циклы, подпрограммы) и поддерживает работу с объектами, их свойствами и методами, за исключением создания новых видов объектов. С встроенным языком 1С тесно связаны язык запросов 1С и язык системы компоновки данных 1С.

4.1. Структура программных модулей, типы данных и базовые операторы

Встроенный язык программирования 1С является также событийно-ориентированным языком. Это означает, что исполнение любых программных модулей конфигурации инициируется неким событием в исполняемой информационной базе. Это может быть открытие формы, запись в регистр, выполнение команды, изменение значения и т.д. Таким образом, весь программный код, создаваемый разработчиком конфигурации, располагается в различных программных модулях объектов конфигураций обычно в составе тех или иных процедур и функций.

Существуют следующие виды программных модулей:

- модули, относящиеся ко всей конфигурации – приложения, сеанса, внешнего соединения;
- общие модули;
- модули общих форм;
- модули, относящиеся к создаваемым объектам метаданных – модули менеджера, объекта, записи, набора записей, форм объектов;
- модуль команды (общей или в составе объекта метаданных).

С программным модулем связано понятие контекста этого модуля, что означает, к каким данным, процедурам и функциям информационной базы име-

ется доступ из процедур и функций этого модуля. Например, контекст модуля формы образуется:

- переменными, описанными внутри исполняемой в данный момент процедуры или функции модуля формы, а также переменными, объявленными в начале модуля вне процедур и функций;
- всеми процедурами и функциями этого же модуля;
- реквизитами формы, к которой относится этот модуль;
- свойствами и методами встроенного языка, относящимся к форме;
- свойствами и методами расширения формы, определяемые типом того объекта, к которому относится форма;
- глобальным контекстом конфигурации, а также общими, в том числе неглобальными, модулями, экспортируемыми функциями и процедурами глобальных общих модулей;
- экспортируемыми переменными, процедурами и функциями модуля приложения.

Сложность запоминания правил образования того или иного контекста компенсируется интеллектуальными подстановками редактором кода по первым введенным буквам имени процедуры, функции, объекта или переменной из доступного контекста, а также сообщениями об ошибках компилятора, который может вызываться как явно, так и автоматически при сохранении программного модуля.

Типы данных 1С образуют достаточно сложную и гибкую систему, в которую могут входить как данные, предусмотренные встроенным языком, так и образующиеся при создании пользователем новых объектов метаданных.

Встроенным языком на уровне платформы предусмотрены так называемые примитивные типы данных (рис. 4.1). К ним относятся *Null*, *Неопределено*, *Число*, *Строка*, *Дата*, *Булево* и *Тип*.

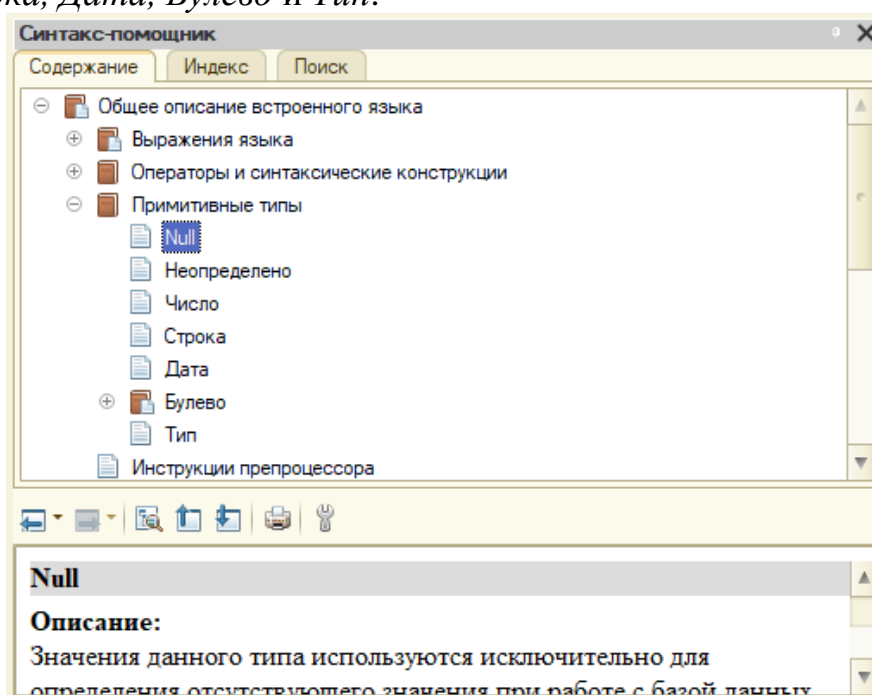


Рис. 4.1. Описание примитивных типов данных в «Синтакс-помощнике»

Для быстрой работы в оперативной памяти компьютера с множествами данных, считанных из хранящихся на дисках таблиц информационной базы или полученных иным способом предназначены универсальные коллекции значений (рис. 4.2). К ним относятся *Массив*, *Структура*, *Соответствие*, *Список значений*, *Таблица значений*, *Дерево значений* и прочие. С их полным перечнем и возможностями также можно ознакомиться в «Синтакс-помощнике».

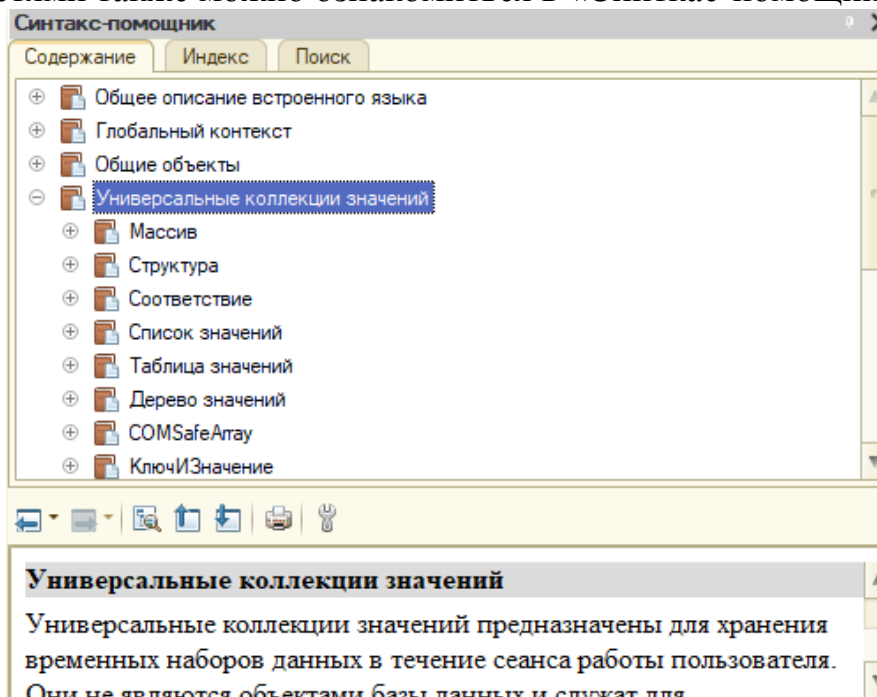


Рис. 4.2. Описание универсальных коллекций в «Синтакс-помощнике»

Следует отметить, что в большинстве языков высокого уровня в синтаксисе самого языка определяется лишь одна коллекция – массив. Встроенный язык 1С предлагает богатый набор готовых коллекций, что сильно упрощает работу программиста, повышает универсальность решения и возможности его дальнейшей поддержки. Сами коллекции значений также могут состоять из значений особых типов. Например, коллекция *Таблица значений* содержит значения типа *СтрокаТаблицыЗначений*.

Также к универсальным (общим) типам данных могут быть отнесены описанные в разделе «Общие объекты» «Синтакс-помощника». Это *ТекстовыйДокумент*, *ТабличныйДокумент*, *Диаграмма*, *ХранилищеЗначений* и прочие. Они используются в различных программных модулях конфигурации для вывода данных либо в каких-либо вспомогательных целях.

К интерфейсным типам относятся форма и её элементы (рис. 4.3). С помощью их методов разработчик может из программных модулей управлять внешним видом приложения в пользовательском режиме.

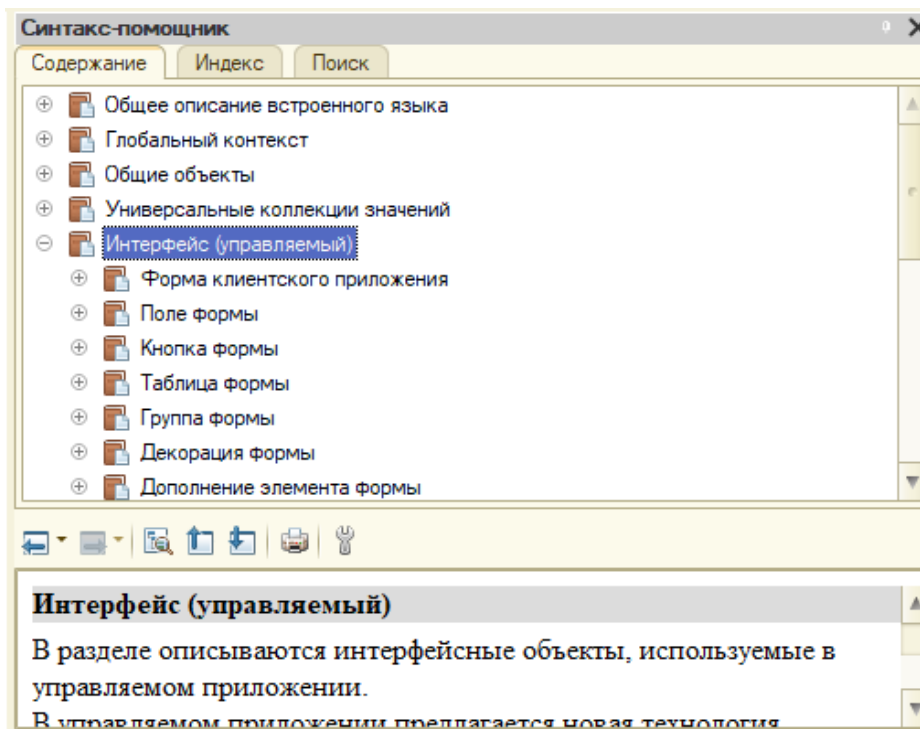


Рис. 4.3. Описание управляемого интерфейса в «Синтакс-помощнике»

Наконец, разработчик в ходе конфигурирования создает новые объекты метаданных (константы, справочники, документы и прочее), называемые прикладными типами данных. Эти типы данных могут фигурировать в программном модуле в качестве ссылки или объекта. Например, у справочника *Справочник1* есть *Реквизит1*, в который может быть записан элемент справочника *Справочник2*. В этом случае тип у реквизита устанавливается *СправочникСсылка.Справочник2*, его программное заполнение может быть реализовано кодом, представленным на рис. 4.4.

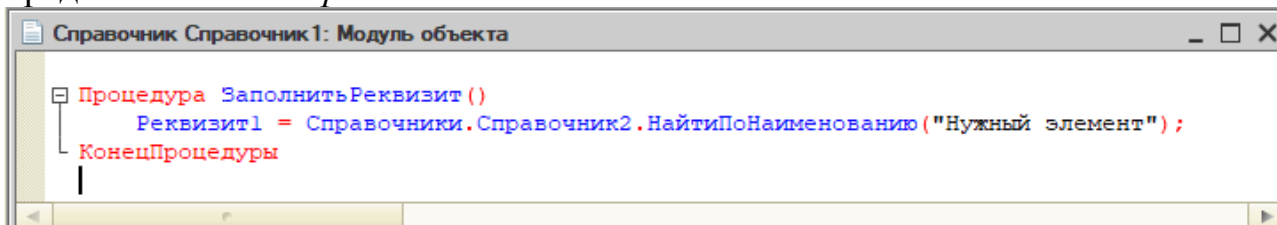


Рис. 4.4. Запись ссылки на элемент справочника в реквизит

Если же необходимо изменить данные экземпляра объекта метаданных (элемента или группы справочника, документа, счета и прочее), то он должен быть представлен в модуле уже не в виде ссылки, а в виде объекта. Например, необходимо переименовать элемент справочника *Справочник2*, хранящийся в реквизите *Реквизит1* (рис. 4.5).

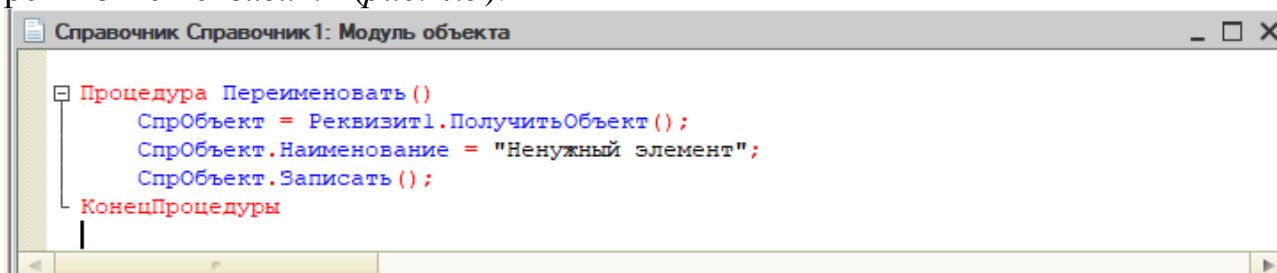


Рис. 4.5. Запись ссылки на элемент справочника в реквизит

В разделе «Выражения языка» «Синтакс-помощника» описаны базовые операции встроенного языка. К ним относятся арифметические (сложение, вычитание, умножение, деление, остаток от деления и унарный минус), конкатенации (соединение строк), логические (операции сравнения). Типы данных явно задаются только в реквизитах объектов конфигурации. Тип переменной определяется только в момент присвоения ей некоторого значения. Это означает, что переменная за время своей жизни в пределах контекста программного модуля может в разные моменты времени хранить значения разных типов. Переменную можно описать в начале процедуры/функции или в начале программного модуля (за исключением общего модуля или модуля менеджера) с помощью ключевого слова *Перем*. Если необходимо, чтобы значение переменной было доступно вне контекста программного модуля, в котором она описана, то в конце строки с описанием добавляют слово *Экспорт* (рис. 4.6).

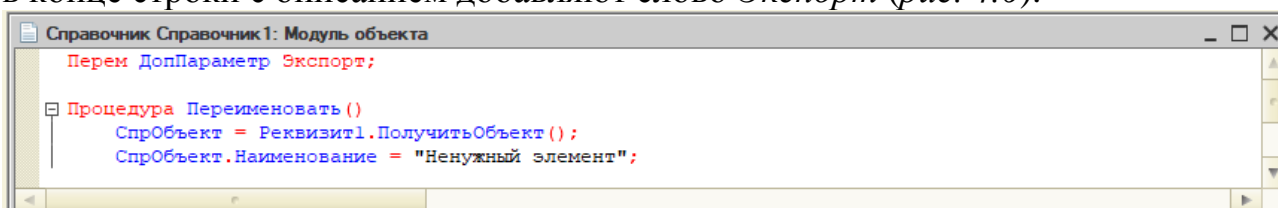


Рис. 4.6. Экспортная переменная *ДопПараметр*, доступная вне контекста модуля

К операторам встроенного языка 1С относятся оператор присваивания, цикла, ветвления, создания нового объекта указанного типа (кроме прикладных) и прочие (рис. 4.7).

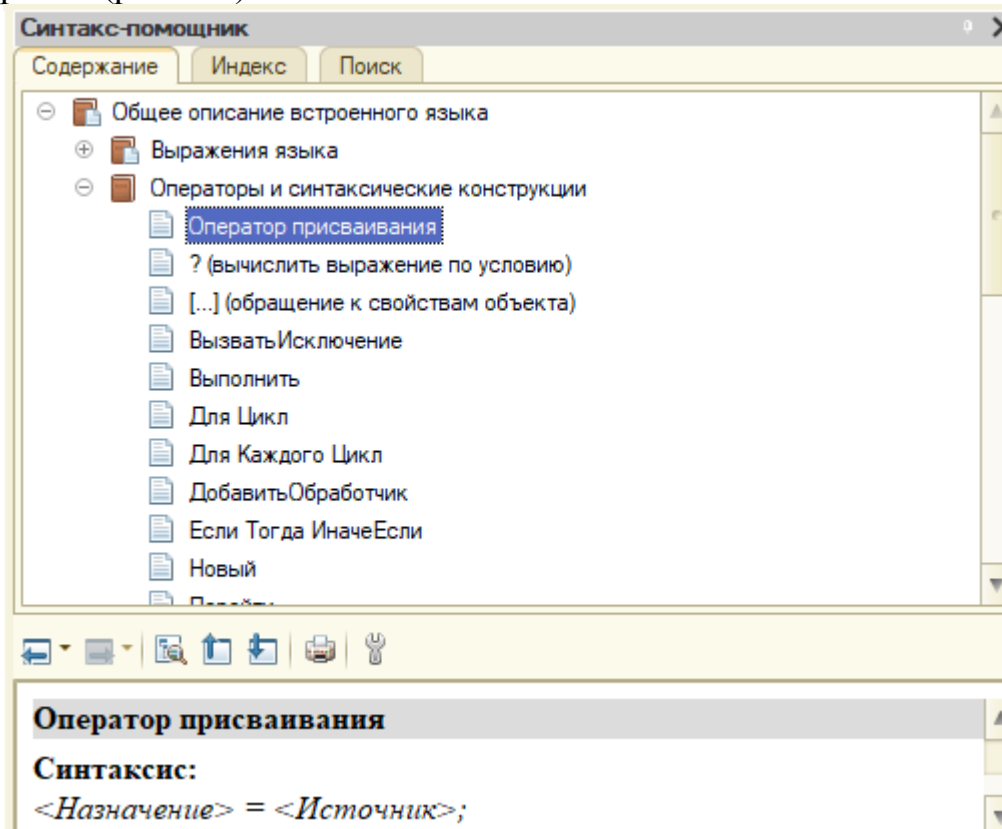


Рис. 4.7. Описание операторов языка в «Синтакс-помощнике»

Следует отметить, что в языке 1С активно используется специальный синтаксис цикла для перебора элементов в универсальных коллекциях.

4.2. Общие и прикладные объекты, универсальные коллекции значений

В силу особой важности следует привести несколько примеров работы с общими, прикладными объектами и универсальными коллекциями значений. Более подробные сведения об использовании этих типов данных и работе с ними можно получить в «Синтакс-помощнике» и примерах программного кода в типовых конфигурациях 1С.

Среди общих объектов, пожалуй, наиболее часто используется *ТабличныйДокумент*. Пример программного кода, использующего этот объект, приведен на рис. 4.8.

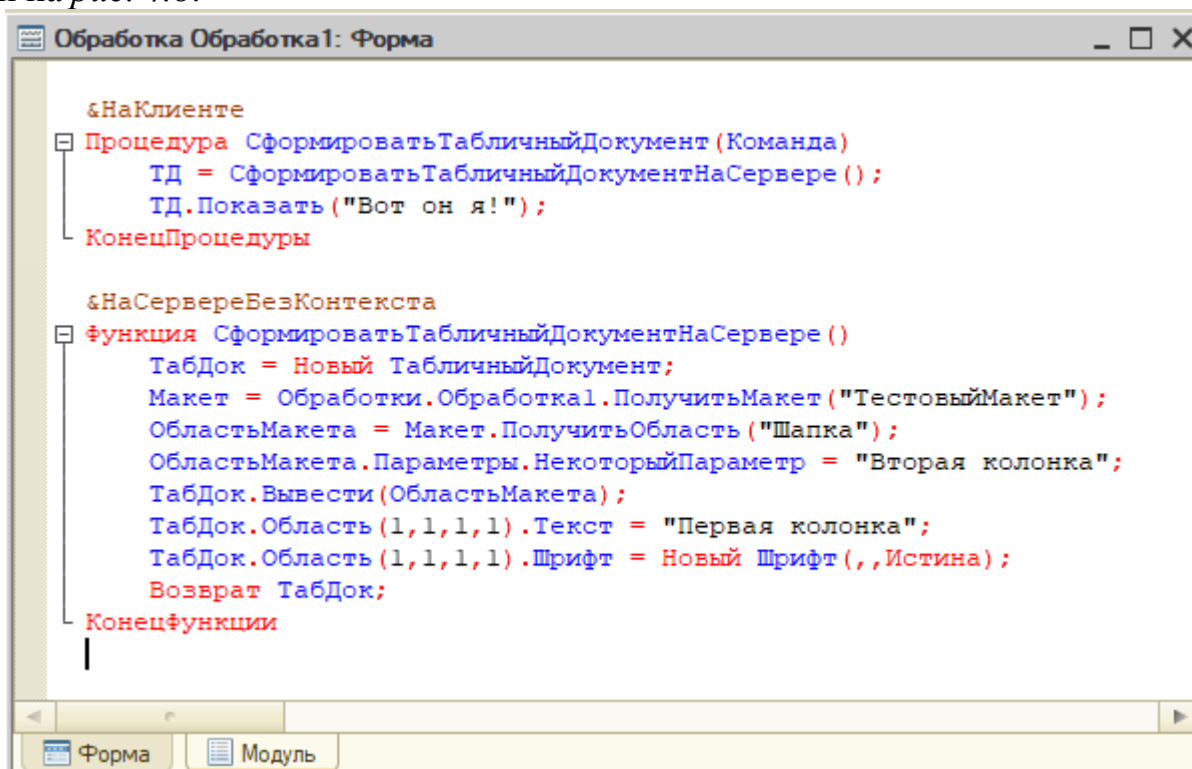


Рис. 4.8. Программное создание и вывод пользователю табличного документа

Процедура *СформироватьТабличныйДокумент* привязана к действию команды (сама команда выведена на форму в виде кнопки). Процедура запускается на клиенте и вызывает серверную функцию *СформироватьТабличныйДокументНаСервере*. В этой функции создается новый объект *ТабличныйДокумент*. Далее из объекта конфигурации *Обработка1* получаем созданный в конфигураторе макет, получаем его именованную область, в которой имеется параметр *НекоторыйПараметр*. Присваиваем ему значение «Вторая колонка», выводим *ОбластьМакета* в наш табличный документ, изменяем текст в ячейке первой колонки первой строки и меняем шрифт этой ячейки на полужирный. Сформированный таким образом табличный документ возвращаем обратно в клиентскую процедуру, где у него вызываем метод *Показать*. Табличный документ отображается пользователю.

Работу с прикладными объектами можно проиллюстрировать на примере обработки, создающей документ *Продажа*, в который включаются все товары, имеющиеся в справочнике *Номенклатура* по одной штуке с ценой 100 рублей.

Документ *Продажа* имеет табличную часть *Товары* с реквизитами *Товар* (тип *СправочникСсылка.Номенклатура*), *Количество* (тип *Число*, 10 разрядов, ноль разрядов после запятой) и *Цена* (тип *Число*, 15 разрядов, 2 после запятой). Программный код модуля формы обработки, формирующий такой документ, приведен на рис. 4.9.

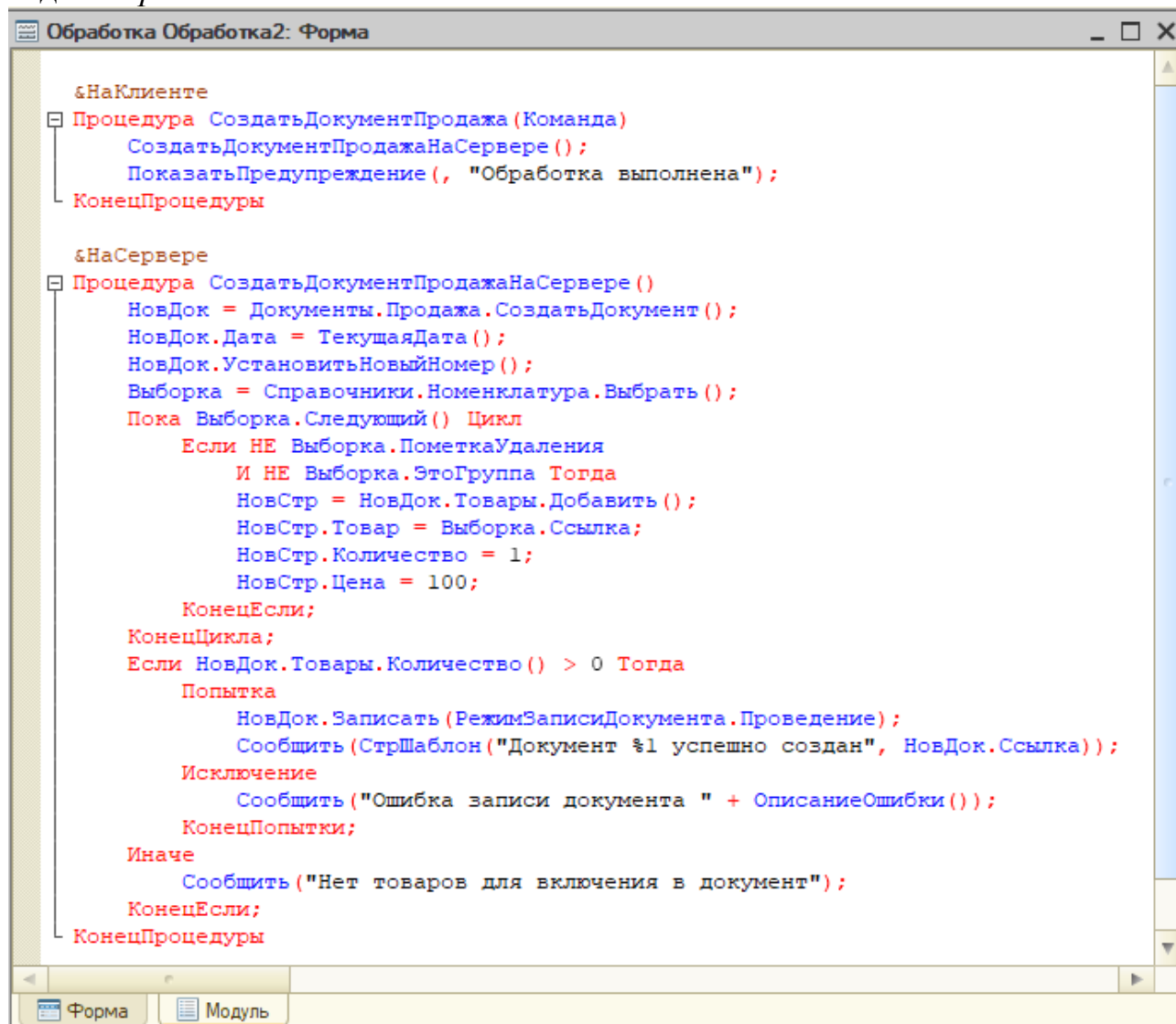


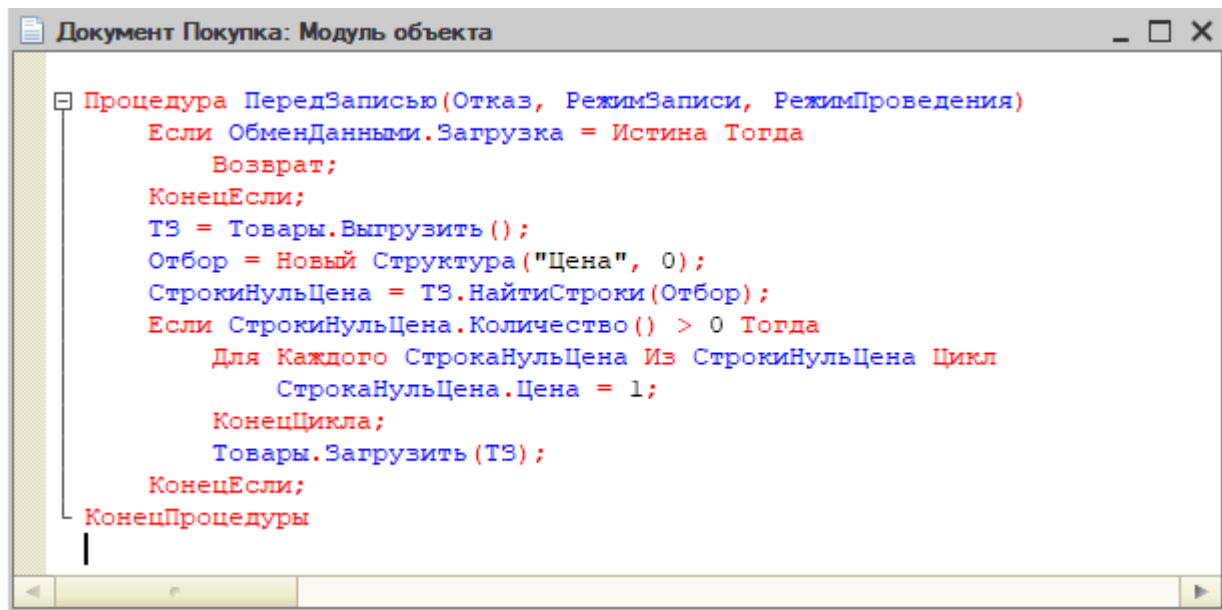
Рис. 4.9. Программное создание и заполнение документа *Продажа*

Клиентская процедура *СоздатьДокументПродажа*, связанная с действием команды, вызывает серверную процедуру *СоздатьДокументПродажаНаСервере*. Создаётся объект нового документа *НовДок*, в стандартный реквизит *Дата* записывается текущий момент времени (реквизиты *Дата* и *Номер* у документа есть всегда). Вызов метода *УстановитьНовыйНомер* необязателен, но имеет смысл, если установлена периодичность номера, а документ создается задним числом. В этом случае значение номера будет зависеть от даты документа. Далее выбираются все записи из справочника *Номенклатура* и осуществляется их обход с помощью метода *Следующий*. Если очередная запись из выборки не помечена на удаление и не является группой справочника, то в табличную часть *Товары* добавляется новая строка, в реквизит *Товар* заносится текущий элемент выборки, устанавливаются значения реквизитов *Количество* и *Цена*. После обхода всей выборки проверяется, есть ли в табличной части до-

кумента хотя бы одна строка. Ведь в справочнике *Номенклатура* может вообще не быть записей либо все они помечены на удаление. Если табличная часть *Товары* заполнена, то в попытке производится запись документа в режиме проведения. Ошибка при записи может возникнуть по разным причинам, например, в связи с отсутствием нужных прав у пользователя или блокировкой таблиц, в которые осуществляется запись, другими пользователями. Запись в попытке позволяет корректно обработать эту ситуацию. Сообщение пользователю может быть сформировано через функцию *СтрШаблон* (предпочтительнее) либо через конкатенацию строк. После выполнения кода на сервере управление передается обратно в клиентскую процедуру, где вызывается процедура *ПоказатьПредупреждение*, которая выбрасывает пользователю диалоговое окно с сообщением, что обработка выполнена. Некоторые обработки в 1С могут выполняться достаточно долго или в фоновом режиме, и поэтому необходимо информировать пользователя о том, что запущенный код выполнен. Подробные сведения о функциях и процедурах встроенного языка для работы с числами, строками, датами и прочее можно получить в «Синтакс-помощнике» в разделе «Глобальный контекст».

Основным инструментом обработки массивов данных в информационной базе является механизм запросов. В случае клиент-серверного варианта использования конфигурации этот механизм обеспечивает максимальное быстродействие и экономию ресурсов серверов и клиентских станций. Однако иногда требуется обработать значительный набор данных непосредственно в оперативной памяти средствами встроенного языка. В этом случае не обойтись без использования универсальных коллекций значений. Наиболее часто используемыми коллекциями являются *Структура* и *ТаблицаЗначений*.

Рассмотрим следующую ситуацию. Существует документ *Покупка*, которым оформляется приход товара от поставщика. Его табличная часть *Товары* аналогична ранее рассмотренной у документа *Продажа*. По требованию функционального заказчика необходимо обеспечить: если пользователь, введивший документ, забыл занести цену товара, то установить её в обязательном порядке равной 1 рублю. Для этого разработчиком было принято решение в момент записи документа найти в табличной части все строки с нулевой ценой и заполнить цену в них единицей. Программный код размещается в процедуре *ПередЗаписью* модуля объекта документа (рис. 4.10). В начале с помощью предопределенного свойства документа *ОбменДанными* проверяется, не является ли данная запись переносом информации из узла распределенной информационной базы или иного источника, что не требует проверок и исправлений документа. Эта конструкция необязательна, но желательна для экономии ресурсов системы при загрузках данных и программном создании элементов справочников и документов. Затем данные из табличной части *Товары* выгружаются в таблицу значений *ТЗ*. Создается структура *Отбор*, в качестве ключа имя колонки поиска, в качестве значения то, что будем искать. Далее с помощью метода *НайтиСтроки* и структуры *Отбор* получаем массив строк, цена в которых нулевая. Обходим строки этого массива и исправляем цену. Таблица значений с исправленными ценами загружается обратно в табличную часть *Товары*.



```
Документ Покупка: Модуль объекта
□ Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)
  Если ОбменДанными.Загрузка = Истина Тогда
    Возврат;
  КонецЕсли;
  ТЗ = Товары.Выгрузить();
  Отбор = Новый Структура("Цена", 0);
  СтрокиНульЦена = ТЗ.НайтиСтроки(Отбор);
  Если СтрокиНульЦена.Количество() > 0 Тогда
    Для Каждого СтрокаНульЦена Из СтрокиНульЦена Цикл
      СтрокаНульЦена.Цена = 1;
    КонецЦикла;
  Товары.Загрузить(ТЗ);
  КонецЕсли;
КонецПроцедуры
```

Рис. 4.10. Заполнение отсутствующей цены перед записью документа

Более подробные сведения о свойствах и методах универсальных коллекций можно найти в «Синтаксис-помощнике». Примеры кода рекомендуется смотреть в типовых конфигурациях 1С.

4.3. Взаимодействие с пользователем, ввод и вывод данных

Взаимодействие программного кода с пользователем системы осуществляется через формы объектов метаданных и общие формы конфигурации. При интерактивной работе пользователя происходят различные события, связанные с открытием либо закрытием формы, воздействием пользователя на элементы формы либо запуск им различных команд, которые отображаются на форме в виде пунктов меню командной панели либо кнопок.

Ввод данных осуществляется через поля ввода формы, которые часто связаны с реквизитами справочника или документа, для которого форма разрабатывается. Вывод данных может осуществляться непосредственно в форму (например форму списка справочника или документа) либо, особенно если необходимо распечатать данные или сохранить их для использования в иных программах, в виде табличного документа, аналогичного листу *Microsoft Excel*.

Рассмотрим программную реализацию интерактивной работы пользователя с формой на примере задачи подбора товаров пользователем из справочника *Номенклатура* в табличную часть *Товары*. В форме документа *Продажа* создается команда, открывающая специальную форму справочника *Номенклатура*, содержащую список справочника и табличную часть, в которую по клику пользователем на строке номенклатуры эта номенклатура будет добавляться в табличную часть формы справочника (рис. 4.11).

Рис. 4.11. Форма для подбора справочника *Номенклатура*

По нажатию на кнопку «Перенести в документ» подобранные товары будут добавлены в табличную часть *Товары* документа *Продажа*. Программный код для реализации данного функционала будет располагаться в модуле формы документа и модуле формы выбора справочника и представлен на *рис. 4.12* и *4.13*.

```

Документ Продажа: ФормаДокумента
  &НаКлиенте
  Процедура ПодборТоваров (Команда)
    ОткрытьФорму ("Справочник.Номенклатура.Форма.ФормаДляПодбора", , Элементы.Товары);
  КонецПроцедуры

  &НаСервере
  Процедура ТоварыОбработкаВыбораНаСервере (АдресХранилища)
    ТЗ = ПолучитьИзВременногоХранилища (АдресХранилища);
    Для Каждого СтрокаПодобранного Из ТЗ Цикл
      НовСтр = Объект.Товары.Добавить ();
      ЗаполнитьЗначенияСвойств (НовСтр, СтрокаПодобранного);
    КонецЦикла;
  КонецПроцедуры

  &НаКлиенте
  Процедура ТоварыОбработкаВыбора (Элемент, ВыбранноеЗначение, СтандартнаяОбработка)
    ТоварыОбработкаВыбораНаСервере (ВыбранноеЗначение);
  КонецПроцедуры
  
```

Рис. 4.12. Код в форме документа *Продажи*

При нажатии пользователя на кнопку в форме вызывается процедура *ПодборТоваров*, которая открывает форму справочника *Номенклатура* для подбора товаров (создана разработчиком) с привязкой к табличной части на форме. Далее в форме для подбора при клике пользователя на строку списка справочника вызывается процедура *СписокВыбор* (рис. 4.13).

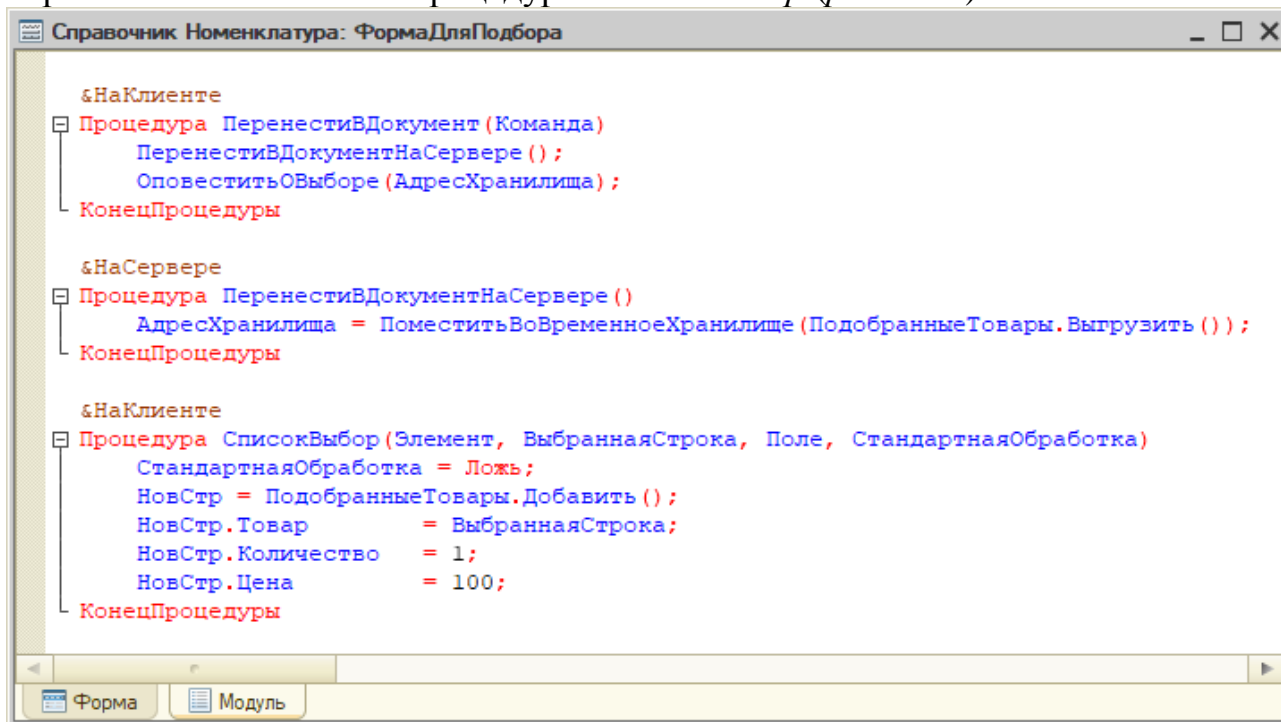


Рис. 4.13. Код в форме справочника *Номенклатура*

СтандартнаяОбработка устанавливается в *Ложь*, чтобы при выборе строки в списке не происходила передача выбранного значения сразу в форму документа. Далее заполняется новая строка в таблице *ПодобранныеТовары*. После завершения пользователем подбора он нажимает в форме на кнопку «Перенести в документ». В серверной процедуре данные из таблицы *ПодобранныеТовары* выгружаются в таблицу значений и помещаются во временное хранилище. Особенность временного хранилища 1С заключается в том, что оно позволяет передавать данные внутри сеанса одного пользователя на сервере между разными пользовательскими формами. Процедура *ОповеститьОВыборе* передает в форму документа адрес таблицы значений с товарами, их количеством и ценой во временном хранилище. Адрес принимает процедура *ТоварыОбработкаВыбора* (рис. 4.12). Она вызывает серверную процедуру *ТоварыОбработкаВыбораНаСервере*, которая получает по переданному адресу таблицу значений из временного хранилища и добавляет строки из этой таблицы в табличную часть *Товары*. Процедура *ЗаполнитьЗначенияСвойств* позволяет заполнить строку одной таблицы по данным строки другой таблицы. Настоятельно рекомендуется ознакомиться с описанием этой процедуры в «Синтаксис-помощнике», так как она значительно упрощает работу с универсальными коллекциями.

Вывод данных пользователю рассмотрим на примере отчета о приобретенных и проданных за заданный период времени товаров. В данном упрощен-

ном варианте отчет будет строиться не по данным регистров накопления, а по данным рассмотренных ранее документов *Покупка* и *Продажа*. Набором данных является запрос (рис. 4.14).

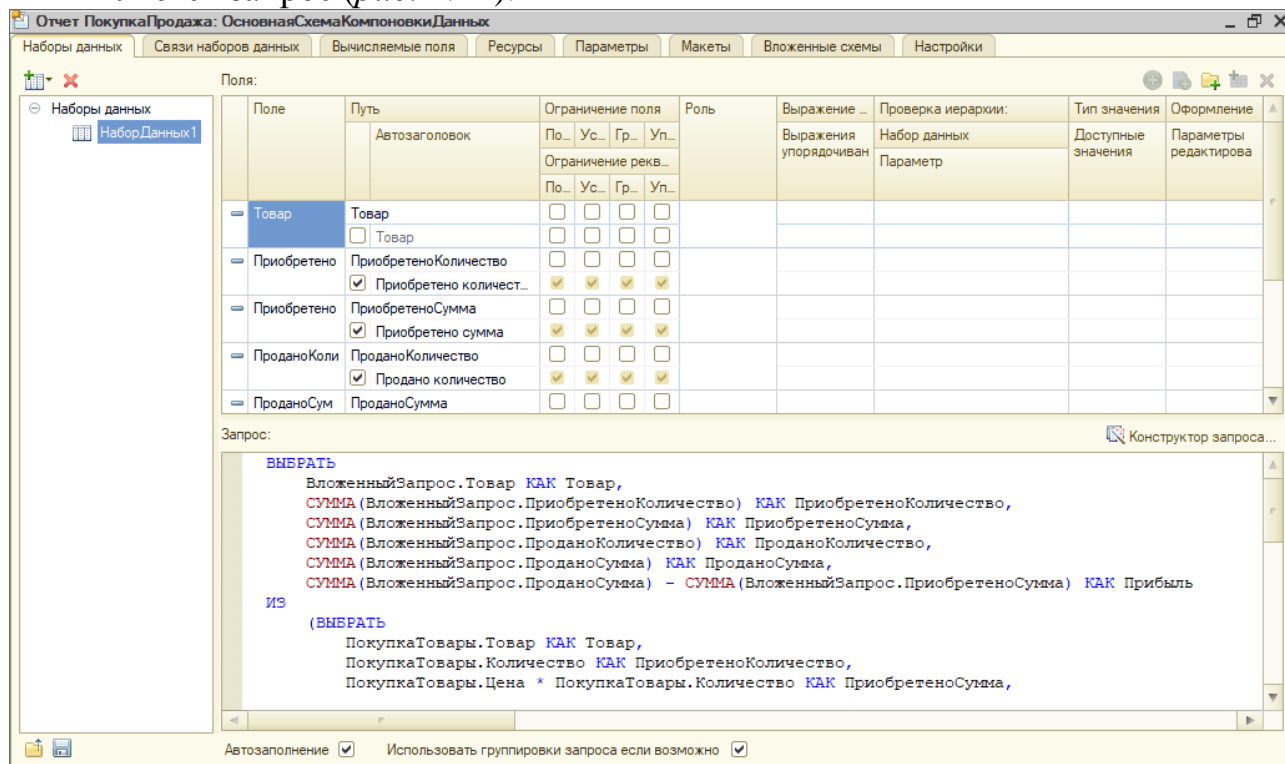


Рис. 4.14. Набор данных СКД

Текст запроса приведен на рис. 4.15.

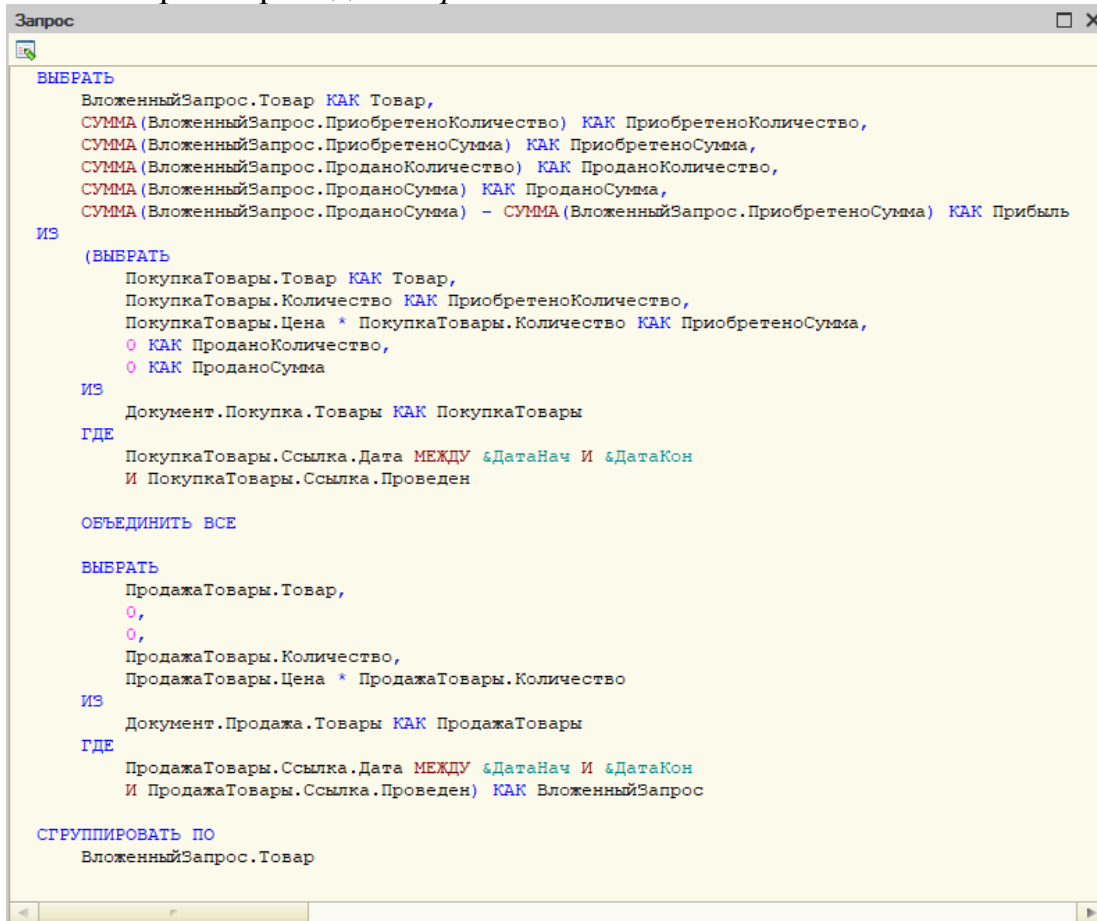


Рис. 4.15. Текст запроса

Вычисляемые поля вынесены в ресурсы (рис. 4.16).

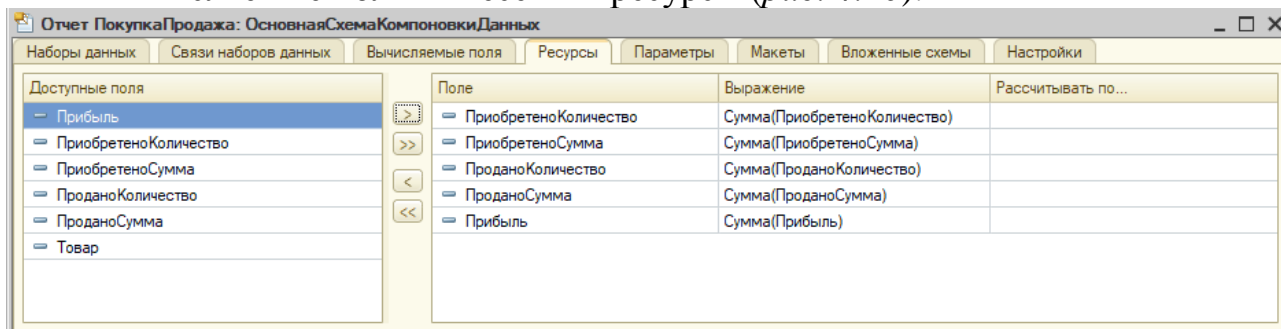


Рис. 4.16. Ресурсы СКД

Для удобства настройки параметров запроса создан параметр отчета Период (рис. 4.17).

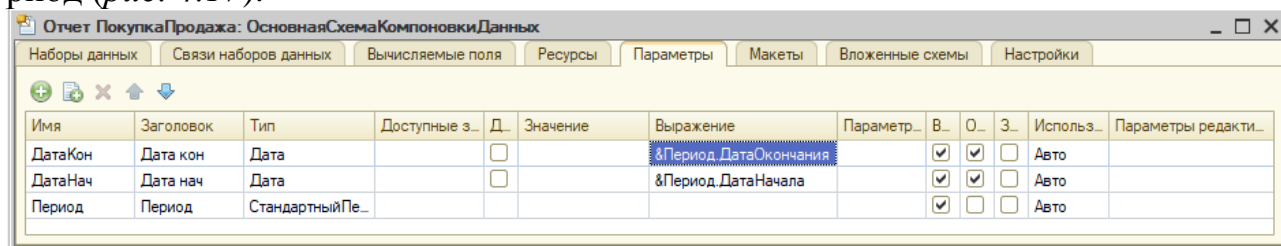


Рис. 4.17. Параметры отчета

В настройках задается структура отчета – группировка по товару и выводимые поля (рис. 4.18).

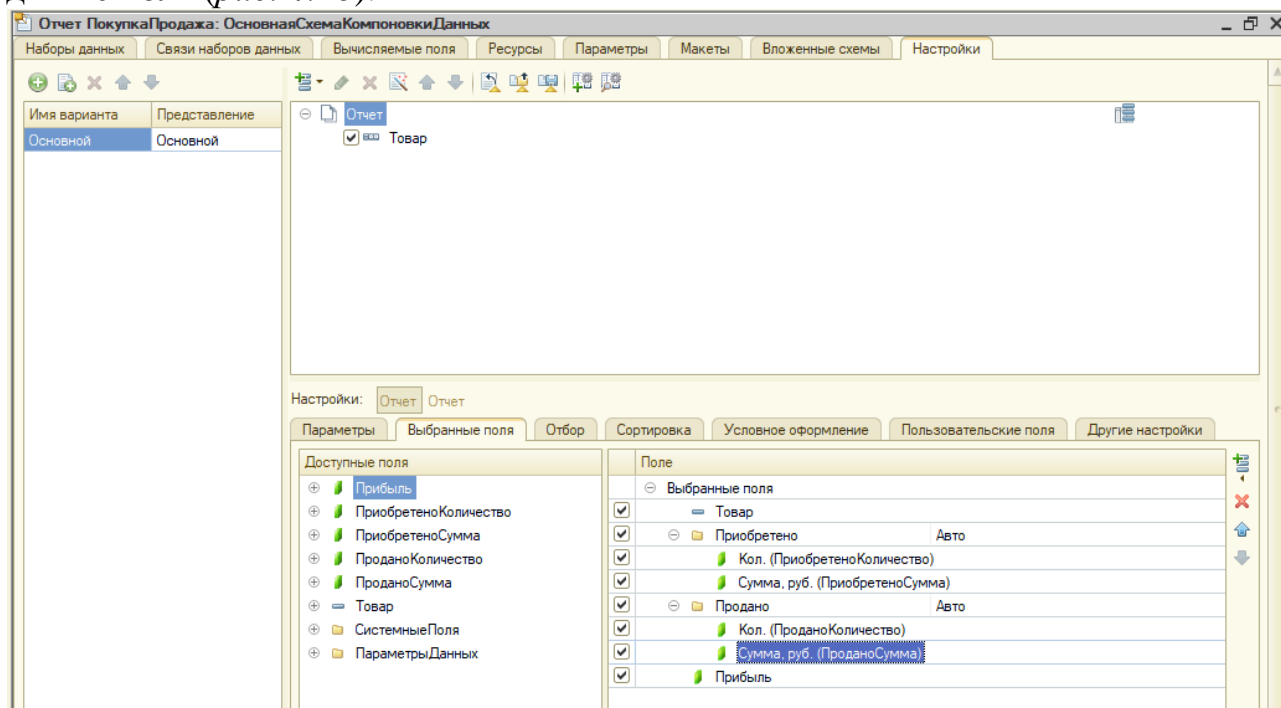


Рис. 4.18. Настройки отчета

Чтобы параметр Период можно было задавать прямо в форме отчета, его необходимо включить в пользовательские настройки (рис. 4.19).

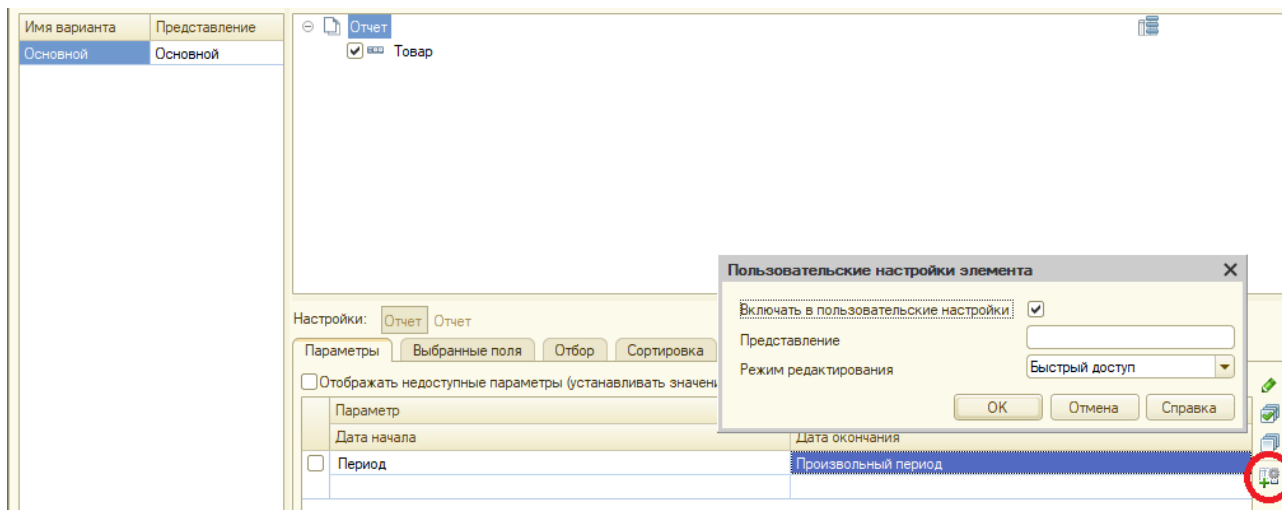


Рис. 4.19. Включение параметра в пользовательские настройки
 Пример работы отчета в режиме «1С:Предприятие» приведен на *рис. 4.20.*

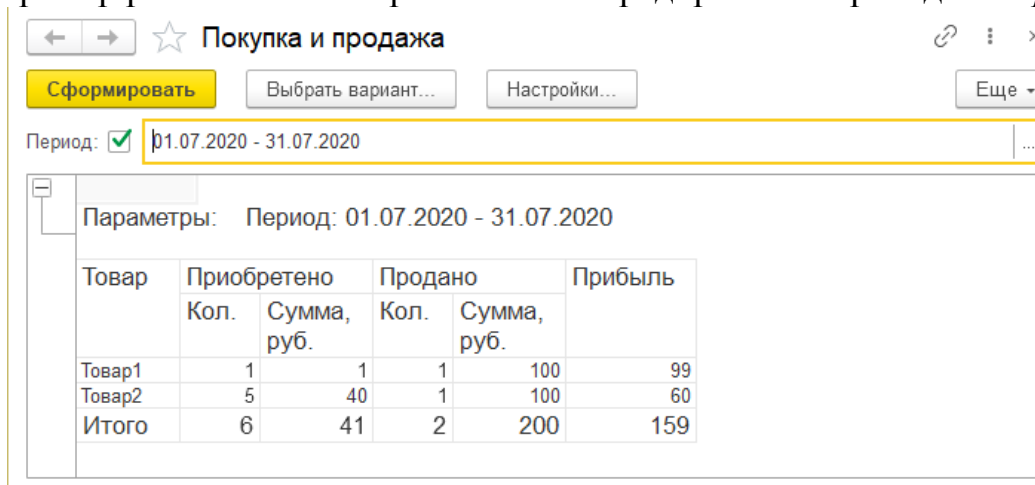


Рис. 4.20. Пример сформированного отчета

Для формирования печатных форм документов используют макеты с именованными областями.

5. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ПЛАТФОРМЕ «1С:ПРЕДПРИЯТИЕ 8.3»

Проектирование и разработку конфигурации информационной базы на платформе «1С:Предприятие 8.3» рассмотрим на примере практической задачи, аналогичной приведенным в разделе 6.

Необходимо разработать конфигурацию для учета лекарственных препаратов в аптеке.

Регистрируются две операции:

- поступление лекарств и прочих товаров в аптеку от поставщиков;
- продажа лекарств и товаров.

При поступлении лекарств и товаров пользователь вводит документ "Поступление товаров", в котором указывает список поступивших лекарств и товаров, их количество и стоимость. В момент поступления нового лекарства, которого нет в базе на момент поступления, пользователь описывает его свойства:

- Форма выпуска;
- Минимальный возраст применения;
- Назначение;
- Действующее вещество;
- Дата государственной регистрации;
- Регистрационный номер;
- Аналог.

Особо подчеркивается, что аналогов может быть больше одного.

При продаже необходимо организовать рабочее место фармацевта, в котором он должен иметь возможность быстро найти нужное лекарство. В верхней части окна фармацевт должен иметь возможность произвести поиск как конкретного медикамента, так и поиск медикаментов по назначению (от головной боли), так и по действующему веществу (парацетамол). В середине окна (на картинке выделено) отображаются найденные медикаменты с указанием цены продажи ("Стоимость" на картинке) и остатка на складе. При необходимости фармацевт переносит мышкой выбранные медикаменты в нижнюю таблицу и указывает количество. Цена и сумма проставляются автоматически и не редактируются пользователем. Последней строкой в нижней таблице отображается итоговая стоимость покупки.

После выбора всех медикаментов, которые нужны покупателю, по нажатию кнопки "Оформить продажу" регистрируется продажа лекарств.

5.1. Разработка архитектуры прикладного решения, создание необходимых констант, справочников и документов

Прежде всего необходимо осмыслить, какие сущности потребуются для реализации данной задачи. Каждая сущность будет выступать объектом в данной предметной области. В данной задаче имеется некоторая организация, осуществляющая розничную торговлю медикаментами. В условии задачи ничего не говорится о необходимости ведения учета в одной информационной базе сразу для нескольких организаций, юридических лиц или частных предпринимателей. Значит, справочник «Организации» можно не создавать. Также в условии задачи не сказано про необходимость реализации бухгалтерского учета, ведения первичной хозяйственной документации по строго регламентированным формам, ведение учета в нескольких валютах и так далее. Так как торговля розничная, то можно вести только учет поставщиков лекарств, а продажа осуществляется обезличенно. Само собой, необходимо хранить информацию о лекарствах, которыми торгует аптека, причем список таких лекарств должен быть всегда под рукой у фармацевта. Необходимо отражать в системе две хозяйственные операции, указанные в задании. При обеих этих операциях необходимо регистрировать количество и стоимость лекарств. Количество обычно измеряется в различных единицах измерения. Необходимо оперативно получать информацию об остатках лекарств в аптеке. Необходимо рассчитывать прибыль от продажи лекарств. Учет взаиморасчетов с поставщиком и отражение оплаты товаров по условию задачи не требуются. Перед разработкой соб-

ственной конфигурации настоятельно рекомендуется ознакомиться с уже имеющимися типовыми решениями 1С.

Переведя вышесказанное на язык метаданных, можно сказать, что необходимо создать справочники «Контрагенты» и «Номенклатура». Подобные названия предпочтительней, чем «Поставщики» и «Лекарства» из-за аналогии с иными конфигурациями 1С. Кроме того, наша аптека может в будущем начать оптовую торговлю, и для выставления счетов и оформления взаиморасчетов понадобится учитывать и покупателей.

Справочник *Контрагенты* будет иерархическим с иерархией групп и элементов, тип кода «Строка», длина наименования 150, без реквизитов и табличных частей. Настоятельно рекомендуется создать все необходимые формы справочника: элемента, группы, списка, выбора, выбора группы.

К справочнику *Номенклатура* предъявляется заданием больше требований. Он также будет иерархическим, длину наименования сделаем нулевой, чтобы вместо наименования использовать реквизит *ПолноеНаименование* (у лекарств бывают очень длинные названия). Это будет строка в 300 символов, реквизит будет использоваться *Для группы и для элемента*. Прочие реквизиты определяются заданием. *ФормаВыпуска – СправочникСсылка.ФормыВыпуска*, неиерархический справочник без реквизитов, длина кода ноль, длина наименования 50. Предполагается, что этот справочник не будет содержать много элементов. Перечислением не делаем его потому, что в будущем могут возникнуть новые формы выпуска препаратов, о которых никто сейчас не знает. *МинимальныйВозрастПрименения* – целое неотрицательное число меньше 100. *Назначение – СправочникСсылка.НазначенияПрепаратов*, аналогичен справочнику *ФормыВыпуска*. *ДействующееВещество – СправочникСсылка.ДействующиеВещества*, похож на номенклатуру, но без реквизитов и табличных частей, наименование длиной 100. *ДатаГосударственнойРегистрации* – дата (без времени).

По требованию задания один и тот же препарат (*Номенклатура*) может иметь несколько аналогов. Аналогами являются такие же препараты, то есть элементы справочника *Номенклатура*. Для хранения множества значений можем использовать табличную часть *Аналоги* с реквизитом *Аналог* (тип *СправочникСсылка.Номенклатура*). Данные справочника *Номенклатура* приведены на рис. 5.1. Для реквизита *ПолноеНаименование* следует указать *Индексировать* (закладка *Использование*), а затем выбрать его на закладке «Поле ввода» вместо *Наименование*. В форме элемента реквизиты располагаем на одной закладке и *Аналоги* на другой. В форме списка выводим реквизиты *Код*, *ПолноеНаименование*, *ДействующееВещество*, *ФормаВыпуска*, *Назначение*, *МинимальныйВозрастПрименения*. Форма выбора группы – *Код*, *ПолноеНаименование*, *ФормаВыпуска*. Форма выбора группы – *Код*, *ПолноеНаименование*.

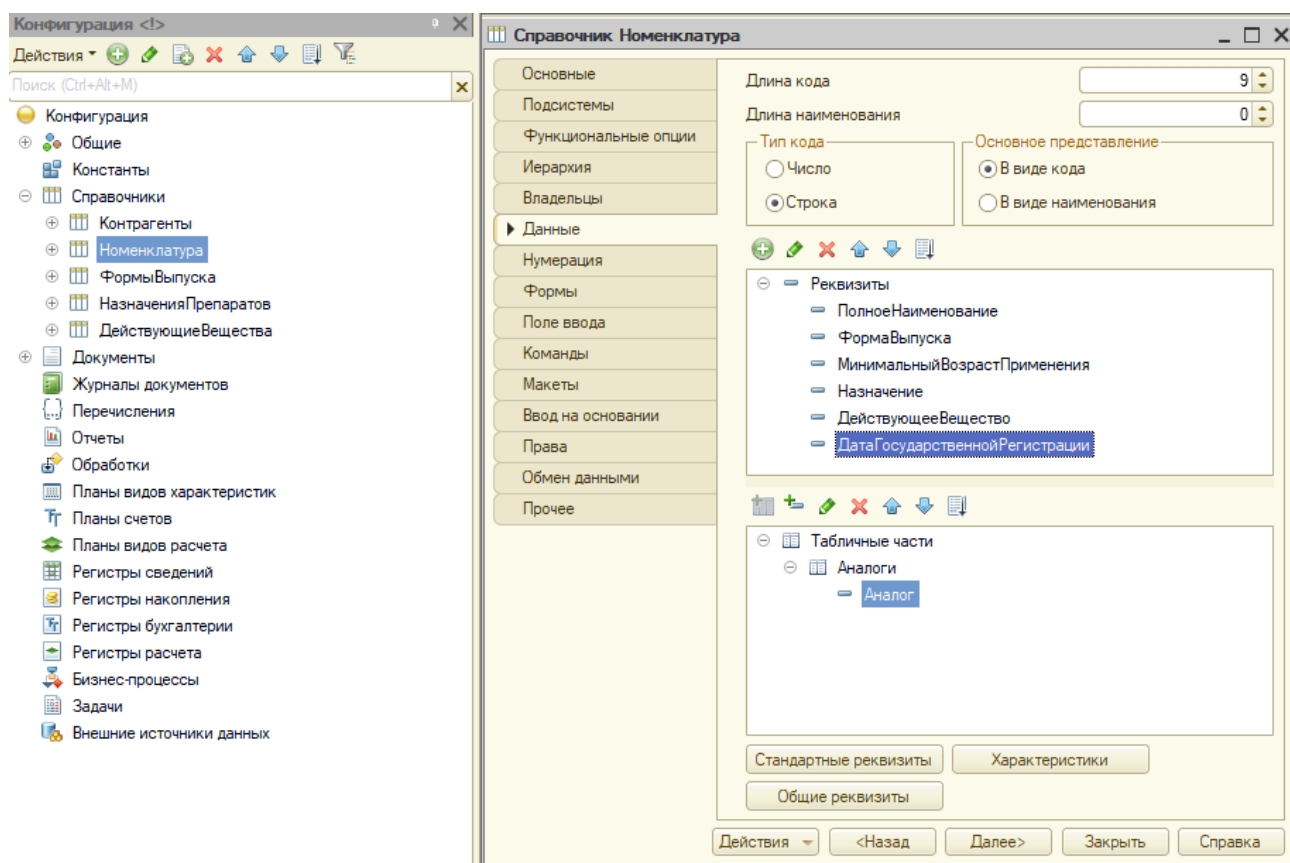


Рис. 5.1. Свойства справочника *Номенклатура*, закладка *Данные*

Чтобы вместо наименования номенклатуры в реквизитах форм отображалось полное наименование, необходимо в модуль менеджера справочника *Номенклатура* добавить следующий код (рис. 5.2).

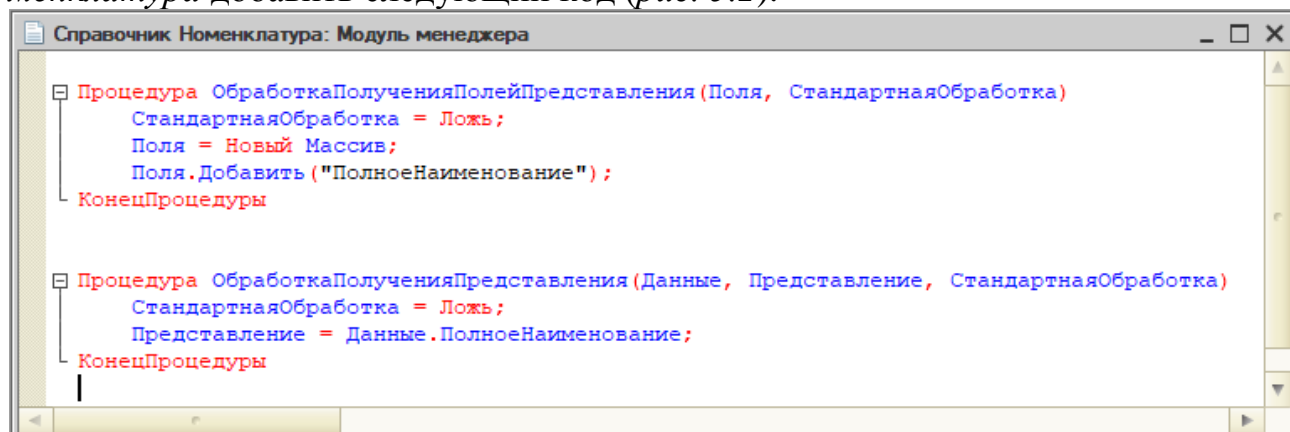


Рис. 5.2. Замена значения представления для справочника

Ещё потребуется справочник *ЕдиницыИзмерения*.

Отражение хозяйственных операций будет осуществляться с помощью двух документов: *Поступление* и *Продажа*.

Документ *Поступление* помимо реквизита *Контрагент* будет содержать табличную часть *Товары* с реквизитами: *Препарат* (тип *СправочникСсылка.Номенклатура*), *Количество* (тип *Число*, 10 разрядов, 3 после запятой), *Цена* (тип *Число*, 15 разрядов, 2 после запятой), *Сумма* (тип *Число*, 15 разрядов, 2 после запятой). Ещё документу необходим реквизит *СуммаДокумента*, в котором будет сохраняться общая стоимость приобретенных лекарств. Так как база

учебная, следовательно, документов в ней будет немного и периодичность номера мы устанавливать не будем.

Создадим формы документа, списка и выбора. В форме списка помимо даты и номера выведем колонки с контрагентом и суммой документа, в форме выбора – дата, номер и контрагент.

Для удобства работы пользователя форма документа нуждается в дополнительном программном коде. Во-первых, необходимо по данным в реквизитах табличной части *Количество* и *Цена* рассчитывать реквизит *Сумма* для строки. Создаем соответствующие обработчики событий и заполняем их программный код в модуле формы (рис. 5.3).

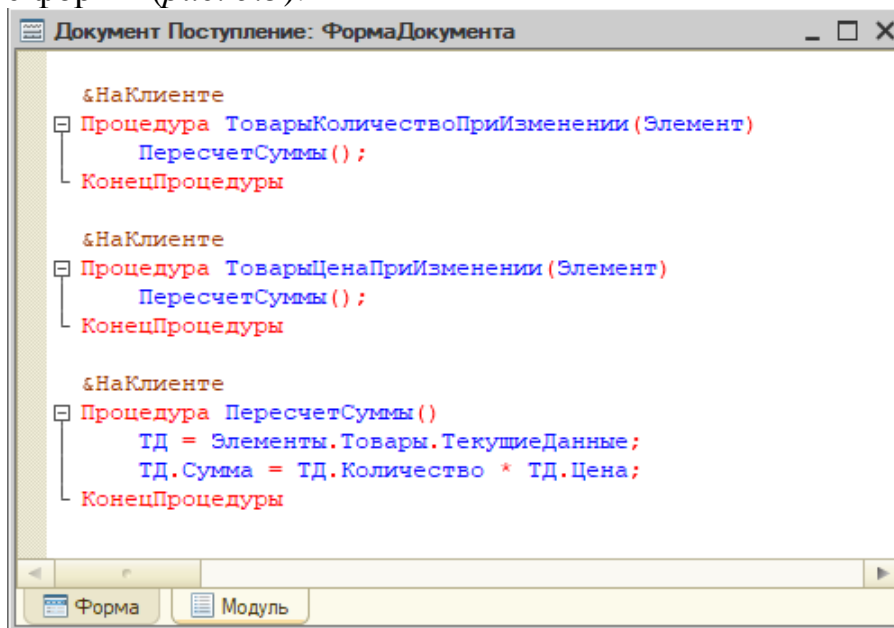


Рис. 5.3. Обработчики изменения значений в *Количество* и *Цена*

Чтобы у пользователя не было соблазна вручную править сумму в строке, установим для этого реквизита *Только просмотр* в свойствах элемента формы. В свойствах элемента формы *Товары* включим использование подвала. Для элемента формы *ТоварыСумма* установим *ПутьКДаннымПодвала* = *Объект.Товары.ИтогСумма*. Теперь сумма документа будет отображаться в подвале табличной части. Реквизит *СуммаДокумента* можно удалить с формы (если он там был).

Во-вторых, итог по колонке табличной части документа отображается, но реквизит *СуммаДокумента* не сохраняется. А значение этого реквизита должно отображаться в форме списка документов, да и в некоторых отчетах может понадобиться. Так как сам реквизит убран с формы и его заменяет значение в подвале табличной части, то оптимально его автоматически пересчитывать в момент, когда пользователь уже закончил работу с формой. Можно поместить соответствующий код в обработчик события формы *ПередЗаписью*, но лучше воспользоваться аналогичной предопределенной процедурой модуля объекта документа. В этом случае пересчет суммы будет происходить всегда при записи документа, даже если запись его осуществлялась программно (рис. 5.4).

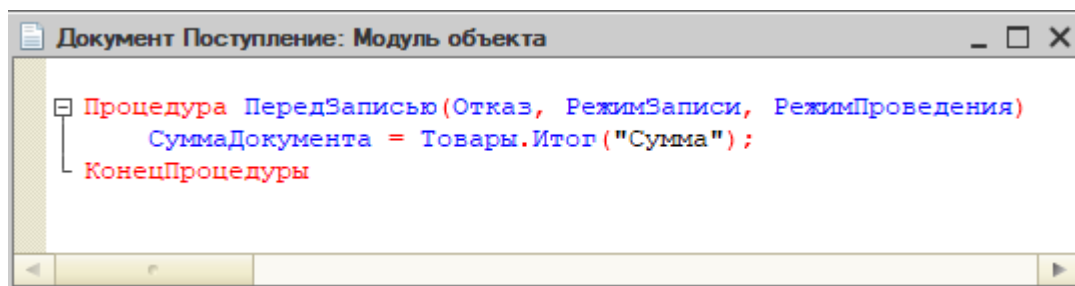


Рис. 5.4. Пересчет суммы документа перед записью

Обычно подобный код в типовых конфигурациях сопровождается проверкой значения *ОбменДанными.Загрузка* (см. раздел 4), но в данной учебной базе пересчет суммы документа будет осуществляться всегда.

В-третьих, необходимо обеспечить обязательность заполнения ряда реквизитов документа. Это необходимо для предотвращения пользователями мусора в базе, когда висят проведенными документы с незаполненным контрагентом, пустой табличной частью товаров и строками, в которых товар забыли указать. Всё это ведет к ошибкам учета и снижает выгоды от автоматизации. Необходимые проверки можно было бы организовать с помощью программного кода, однако имеются удобные механизмы платформы, позволяющие проводить необходимые проверки с выдачей сообщений о некорректности данных в момент проведения документа. Например, установим выдачу ошибки при проведении документа, если не заполнен реквизит *Контрагент* (рис.5.5).

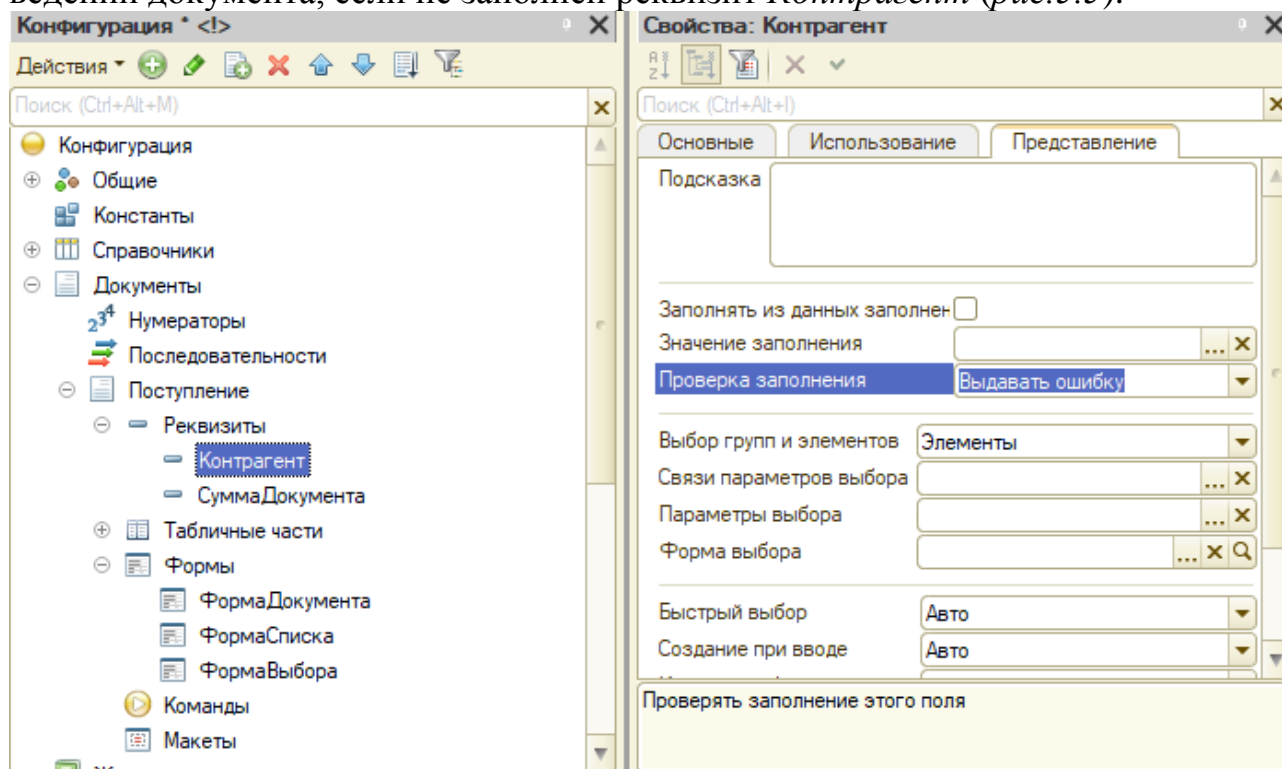


Рис. 5.5. Проверка заполнения реквизита *Контрагент*

Аналогичным образом запретим пустую табличную часть *Товары* (без строк) и пустое значение реквизитов *Препарат* и *Количество*. Цену и сумму проверять не будем, так как наша аптека будет иногда бедным людям отдавать лекарства даром.

Документ *Продажа* будет полностью (на данном этапе разработки) аналогичен документу *Поступление* за исключением реквизита *Контрагент*, который в нем не нужен. Создание этого документа можно выполнить копированием с последующим редактированием.

Отпуск товаров не только в аптеки, но и в любой розничной торговой точке осуществляется по некоторым фиксированным (в течение определенного промежутка времени) ценам, устанавливаемым руководством организации. Конечно, провизор может вбивать цены в документ при продаже из бумажки с распечатанным прайс-листом. Но правильнее, если эти данные хранятся в информационной базе и подставляются в документы автоматически. В учебной базе в упрощенном варианте для хранения отпускных цен может использоваться периодический регистр сведений *ЦеныНоменклатуры*. Периодичность установим в пределах дня, режим записи независимый (хотя в типовых конфигурациях используется специальный документ «Установка цен номенклатуры»), измерение *Номенклатура* (тип *СправочникСсылка.Номенклатура*), ресурс *Цена* (тип *Число*, 15 разрядов, 2 после запятой). Формы записи и формы списка создадим.

По условиям задачи подстановка цен и формирование документа *Продажа* должны происходить из рабочего места провизора. Это будет рассмотрено позже. Однако в учебных целях реализуем подстановку цены в табличную часть *Товары* документа *Продажа* при выборе препарата в обработчике *ПриИзменении* (рис. 5.6).

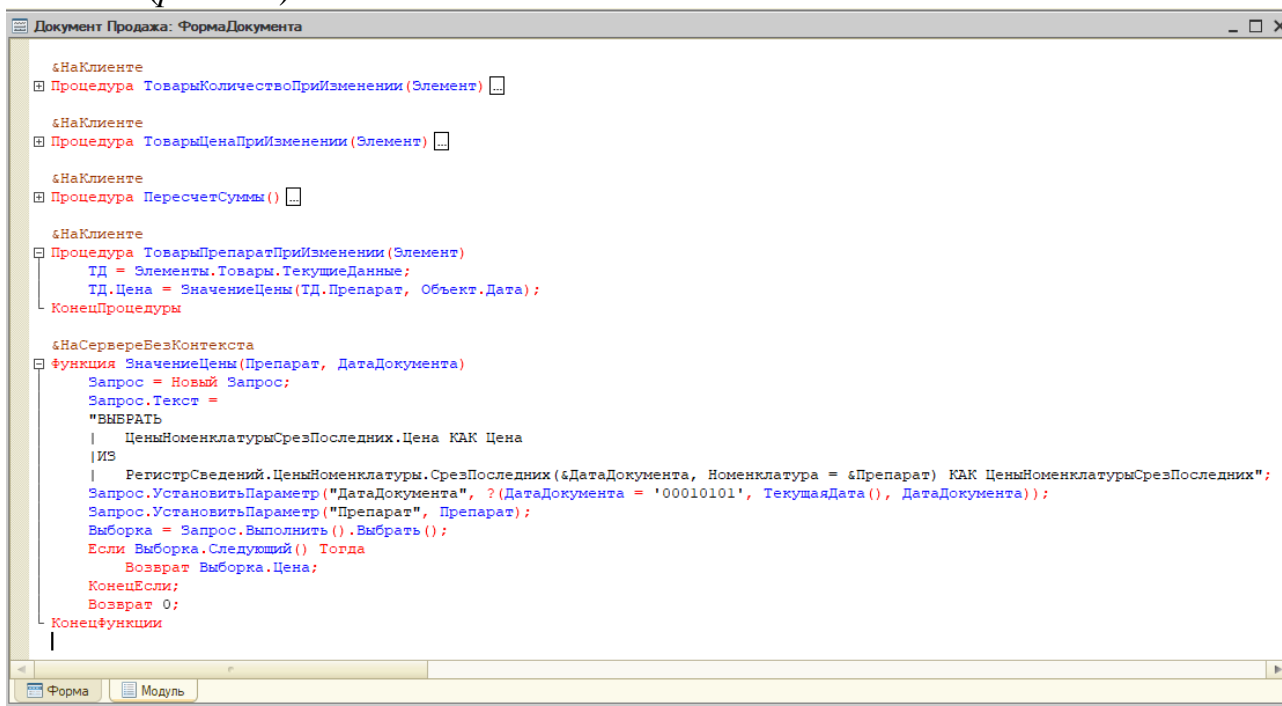


Рис. 5.6. Заполнение цены по данным регистра сведений

Реализация получения данных из информационной базы через запрос всегда предпочтительней, чем через обращение к методам менеджеров объектов конфигурации (например, через *РегистрыСведений.ЦеныНоменклатуры.СрезПоследних(...)*), так как, несмотря на большее ко-

личество строк на встроенном языке 1С, программный код выполняется намного быстрее.

В типовых конфигурациях используются несколько категорий цен.

Название организации разместим в константе *НаименованиеОрганизации* (тип *Строка*, неограниченная длина).

5.2. Создание регистров накопления

Как уже неоднократно говорилось в курсе изучения 1С, данные можно было бы хранить и обрабатывать непосредственно во введенных документах. Однако для повышения быстродействия системы и удобства построения аналитических отчетов оперативные данные документов также необходимо хранить в регистрах накопления, бухгалтерии либо расчетов. Выбор типов регистров определяется особенностями и постановкой задачи. В данном случае нет необходимости вести бухгалтерский учет либо выполнять сложные периодические расчеты, поэтому используем самые простые регистры накопления: *остатков* либо *оборотов*. Необходимо вести учет текущего остатка препаратов и прибыли (отпускная стоимость минус себестоимость) от каждой продажи каждого препарата. Для упрощения учебной задачи будем считать, что используется средневзвешенная себестоимость (порядок расчета себестоимости прописан в учетной политике организации). В этом случае достаточно учитывать остаток каждого препарата в количественном и суммовом выражении. Для этого используем регистр остатков *ОстаткиНоменклатуры* с измерением *Номенклатура* (тип *СправочникСсылка.Номенклатура*) и ресурсами *Количество* (тип *Число*, 10 разрядов, 3 после запятой) и *Стоимость* (тип *Число*, 15 разрядов, 2 после запятой). На закладке «Основные» свойств регистра выбираем вид регистра *Остатки*. В качестве регистраторов будут выступать оба наших документа: *Поступление* будет увеличивать остаток, *Продажа* его уменьшать. Для реализации движений документов в обработке проведения воспользуемся не конструктором движений, а более быстрым и правильным механизмом подготовки записей с помощью запросов и их загрузки в таблицы движений документа. Текст процедуры *ОбработкаПроведения* в модуле объекта *Поступление* приведен на *рис. 5.7*. Сначала с помощью запроса подготавливается набор записей регистра, затем этот набор записей помещается в соответствующую таблицу движений и устанавливается признак необходимости записи. В результате в регистре увеличивается остаток и в количественном, и в суммовом выражении.

```

Документ Поступление: Модуль объекта
□ Процедура ОбработкаПроведения(Отказ, РежимПроведения)
  Запрос = Новый Запрос;
  Запрос.Текст =
  "ВЫБРАТЬ
  |   ЗНАЧЕНИЕ(ВидДвиженияНакопления.Приход) КАК ВидДвижения,
  |   ПоступлениеТовары.Ссылка.Ссылка КАК Регистратор,
  |   ПоступлениеТовары.Ссылка.Дата КАК Период,
  |   ПоступлениеТовары.Препарат КАК Номенклатура,
  |   СУММА(ПоступлениеТовары.Количество) КАК Количество,
  |   СУММА(ПоступлениеТовары.Сумма) КАК Стоимость
  |ИЗ
  |   Документ.Поступление.Товары КАК ПоступлениеТовары
  |ГДЕ
  |   ПоступлениеТовары.Ссылка = &Ссылка
  |
  |СГРУППИРОВАТЬ ПО
  |   ПоступлениеТовары.Ссылка.Ссылка,
  |   ПоступлениеТовары.Ссылка.Дата,
  |   ПоступлениеТовары.Препарат";
  Запрос.УстановитьПараметр("Ссылка", Ссылка);
  Движения.ОстаткиНоменклатуры.Загрузить(Запрос.Выполнить().Выгрузить());
  Движения.ОстаткиНоменклатуры.Записывать = Истина;
КонецПроцедуры

```

Рис. 5.7. Обработка проведения документа *Поступление*

Документ *Продажа* будет делать движения сразу по двум регистрам: регистру *ОстаткиНоменклатуры* (но с видом движения *Расход*) и оборотному регистру *ПродажиНоменклатуры*, который будет хранить информацию о проданных товарах, их себестоимости и продажной стоимости. На закладке «Основные» свойств регистра *ПродажиНоменклатуры* выбираем вид регистра *Обороты*. Измерение *Номенклатура* (тип *СправочникСсылка.Номенклатура*), ресурсы – *Количество* (тип *Число*, 10 разрядов, 3 после запятой), *Себестоимость* (тип *Число*, 15 разрядов, 2 после запятой) и *СтоимостьПродажи* (тип *Число*, 15 разрядов, 2 после запятой). Регистратор на закладке «Движения» - документ *Продажа*.

Обработка проведения документа *Продажа* имеет более сложный код. Во-первых, необходимо выполнить уменьшение количества и стоимости в регистре *ОстаткиНоменклатуры*. Расход количества соответствует количеству проданного товара, а вот расход стоимости вычисляется по остатку стоимости товара на момент продажи и остатку количества товара на тот же момент

$$\text{СуммаСписания} = \text{ОстатокСуммы} / \text{ОстатокКоличества} * \text{СколькоСписать}$$

При этом следует учесть проблемы округления. Не должно из-за него происходить полного списания количества в ноль при ненулевом остатке стоимости. Сумма списания будет соответствовать движению по ресурсу *Себестоимость* регистра *ПродажиНоменклатуры*, а сумма строки документа *Продажа* – движению по ресурсу *СтоимостьПродажи*. Ниже приводится текст пакета запросов, подготавливающих необходимые таблицы движения.

```

"ВЫБРАТЬ
|   ПоступлениеТовары.Препарат КАК Номенклатура,
|   СУММА(ПоступлениеТовары.Количество) КАК Количество,
|   СУММА(ПоступлениеТовары.Сумма) КАК Стоимость

```

```

ПОМЕСТИТЬ ВтПродажа
ИЗ
    Документ.Поступление.Товары КАК ПоступлениеТовары
ГДЕ
    ПоступлениеТовары.Ссылка = &Ссылка
СГРУППИРОВАТЬ ПО
    ПоступлениеТовары.Препарат
;

////////////////////////////////////
ВЫБРАТЬ
    ОстаткиНоменклатуры.Остатки.Номенклатура КАК Номенклатура,
    ОстаткиНоменклатуры.Остатки.КоличествоОстаток КАК КоличествоОстаток,
    ОстаткиНоменклатуры.Остатки.СтоимостьОстаток КАК СтоимостьОстаток
ПОМЕСТИТЬ ВтОстатки
ИЗ
    РегистрНакопления.ОстаткиНоменклатуры.Остатки(
        &Период,
        Номенклатура В
            (ВЫБРАТЬ
                ИЗ
                    Вт.Номенклатура
                    ВтПродажа КАК Вт)) КАК ОстаткиНоменклатуры.Остатки
;

////////////////////////////////////
ВЫБРАТЬ
    ВтПродажа.Номенклатура КАК Номенклатура,
    ВтПродажа.Количество КАК Количество,
    ВЫБОР
        КОГДА ВтОстатки.КоличествоОстаток ЕСТЬ NULL
            ТОГДА 0
        КОГДА ВтОстатки.КоличествоОстаток <= ВтПродажа.Количество
            ТОГДА ВтОстатки.СтоимостьОстаток
        ИНАЧЕ ВтПродажа.Количество * ВтОстатки.СтоимостьОстаток / ВтОстатки.КоличествоОстаток
    КОНЕЦ КАК Себестоимость,
    ВтПродажа.Стоимость КАК СтоимостьПродажи
ПОМЕСТИТЬ ВтСводная
ИЗ
    ВтПродажа КАК ВтПродажа
    ЛЕВОЕ СОЕДИНЕНИЕ ВтОстатки КАК ВтОстатки
    ПО ВтПродажа.Номенклатура = ВтОстатки.Номенклатура
;

////////////////////////////////////
УНИЧТОЖИТЬ ВтПродажа
;

////////////////////////////////////
УНИЧТОЖИТЬ ВтОстатки
;

////////////////////////////////////
ВЫБРАТЬ
    ЗНАЧЕНИЕ(ВидДвиженияНакопления.Расход) КАК ВидДвижения,
    &Ссылка КАК Регистратор,
    &Период КАК Период,
    ВтСводная.Номенклатура КАК Номенклатура,
    ВтСводная.Количество КАК Количество,
    ВтСводная.Себестоимость КАК Стоимость
ИЗ
    ВтСводная КАК ВтСводная
;

////////////////////////////////////
ВЫБРАТЬ
    &Ссылка КАК Регистратор,
    &Период КАК Период,
    ВтСводная.Номенклатура КАК Номенклатура,
    ВтСводная.Количество КАК Количество,
    ВтСводная.Себестоимость КАК Себестоимость,
    ВтСводная.СтоимостьПродажи КАК СтоимостьПродажи
ИЗ
    ВтСводная КАК ВтСводная";

```

Вначале формируются три временные таблицы, собирающие данные из документа и из регистра. Окончательной является таблица *ВтСводная*. Данные

из неё распределяются по регистрам *ОстаткиНоменклатуры* и *ПродажиНоменклатуры*. Для этого используются два последних запроса **ВЫБРАТЬ**. Метод запроса *ВыполнитьПакет* возвращает массив результатов. Данные двух последних результатов запроса загружаются в соответствующие таблицы движений. Принудительно вызывается метод *Записать* для движений документа (рис. 5.8).

```

Документ Продажа: Модуль объекта
□ Процедура ОбработкаПроведения(Отказ, РежимПроведения)
    Запрос = Новый Запрос;
    Запрос.Текст = ТекстЗапроса();
    Запрос.УстановитьПараметр("Ссылка", Ссылка);
    Запрос.УстановитьПараметр("Период", Дата);
    Результаты = Запрос.ВыполнитьПакет();
    РезультатОстатки = Результаты[Результаты.ВГраница()-1];
    РезультатПродажи = Результаты[Результаты.ВГраница()];
    Движения.ОстаткиНоменклатуры.Загрузить(РезультатОстатки.Выгрузить());
    Движения.ОстаткиНоменклатуры.Записывать = Истина;
    Движения.ПродажиНоменклатуры.Загрузить(РезультатПродажи.Выгрузить());
    Движения.ПродажиНоменклатуры.Записывать = Истина;
    Движения.Записать();
    Если РежимПроведения = РежимПроведенияДокумента.Оперативный Тогда
        КонтрольРезультатовПроведения(Отказ);
    КонецЕсли;
КонецПроцедуры

□ Функция ТекстЗапроса()
    Возврат
    "ВЫБРАТЬ
    |     ПродажаТовары.Препарат КАК Номенклатура,

```

Рис. 5.8. Обработка проведения документа *Продажа*

В случае перепроведения документа, чтобы его движения не влияли на результаты расчета, в свойствах на закладке «Движения» для удаления движений выбрано *Удалять автоматически*.

В конце обработки проведения для случая, когда осуществляется оперативное проведение документа, выполняется проверка, не ушли ли остатки в минус. Аналогичный метод проверки (после выполнения движений документом, а не до) используется в современных типовых конфигурациях. Код проверки вынесен в отдельную процедуру *КонтрольРезультатовПроведения* (рис. 5.9).

```

Документ Продажа: Модуль объекта
[+] функция ТекстЗапроса ()
[+] Процедура КонтрольРезультатовПроведения (Отказ)
    Запрос = Новый Запрос;
    Запрос.Текст =
        "ВЫВРАТЬ
        | ОстаткиНоменклатурыОстатки.Номенклатура КАК Номенклатура,
        | ОстаткиНоменклатурыОстатки.КоличествоОстаток КАК КоличествоОстаток
        |ИЗ
        | РегистрНакопления.ОстаткиНоменклатуры.Остатки (
        |
        |         Номенклатура В
        |         (ВЫВРАТЬ
        |         Товары.Препарат
        |         ИЗ
        |         Документ.Продажа.Товары КАК Товары
        |         ГДЕ
        |         Товары.Ссылка = &Ссылка)) КАК ОстаткиНоменклатурыОстатки
        |ГДЕ
        | ОстаткиНоменклатурыОстатки.КоличествоОстаток < 0";
    Запрос.УстановитьПараметр ("Ссылка", Ссылка);
    Результат = Запрос.Выполнить ();
    Если Результат.Пустой () Тогда
        Возврат;
    КонецЕсли;
    Выборка = Результат.Выбрать ();
    Пока Выборка.Следующий () Цикл
        Сообщить (СтрШаблон ("Количество проданного %1 превышает остаток на складе на %2",
        Выборка.Номенклатура,
        -Выборка.КоличествоОстаток));
    КонецЦикла;
    Отказ = Истина;
КонецПроцедуры

```

Рис. 5.9. Контроль регистров по результатам оперативного проведения

Если в результате проведения документа остатки по регистру *Остатки-Номенклатуры* ушли в минус, то пользователю выводится сообщение о том, сколько и какого товара не хватает, а у переменной *Отказ* устанавливается значение *Истина*, что означает запрет проведения документа со стороны технологической платформы. В режиме «1С:Предприятие» в этом случае пользователю сначала выводится системное диалоговое окно, а затем в нижней части окна приложения – сообщения о товарах, которых недостаточно для продажи. Документ при этом с новыми данными не записывается.

5.3. Создание печатных форм и отчетов

Наиболее актуальной печатной формой в любой хозяйственной деятельности является накладная. В накладной перечисляется поставляемый товар, его количество, единица измерения этого количества, цена и сумма. Эта информация отображается в виде таблицы, которая занимает основную часть документа Накладная. Выше таблицы располагается шапка, ниже – подвал. В шапке указывают название документа, его номер, дату (без времени), поставщика и покупателя. В подвале – общую сумму документа и места для подписей того, кто отпустил товар, и того, кто его получил. Правительство Российской Федерации и подчиненные ему государственные органы устанавливают более жесткие требования к первичным документам товародвижения (что отражено в типовых

конфигурациях 1С), но для учебной задачи достаточно формировать упрощенную печатную форму.

Будем предполагать, что создаваемый макет накладной будет использоваться как для документа *Продажа*, так и для документа *Покупка*. В последнем случае это также бывает актуально в практической хозяйственной деятельности небольших организаций, когда накладную на поступление иногда приходится печатать за поставщика. Поэтому будем создавать единый макет, который разместим в ветке *Общие макеты* раздела *Общие конфигурации* (рис. 5.10).

	1	2	3	4	5	6
1						
Шапка	2					
	3	<Накладная №[Номер] от [ДатаДок]>				
	4	<Поставщик: [Поставщик]>				
	5					
	6	<Покупатель: [Покупатель]>				
	7					
	8	П/п	Препарат	Ед.изм.	Кол.	Цена, руб Сумма, руб.
	9					
СтрТаб	10	<Ном	<Препарат>	<ЕдИзм>	<Кол>	<Цена> <Сумма>
	11					
Подвал	12					
	13	Итого сумма:			<СуммаДокумента>	
	14					
	15					
	16	Отпустил _____		Получил _____		
	17					
	18					
	19					

Рис. 5.10. Макет накладной

Для формирования накладной нам понадобится информация об единице измерения количества в накладной и название нашей организации (аптеки). Для упрощения учебной задачи будем предполагать, что каждый препарат учитывается только в одной единице измерения (штука, упаковка, флакон и так далее). Поэтому добавим в справочник *Номенклатура* реквизит *ЕдиницаИзмерения* (тип *СправочникСсылка.ЕдиницыИзмерения*). Название организации разместим в константе *НаименованиеОрганизации*.

Программный код для формирования печатной формы также будет весьма похож для обоих документов. Поэтому разместим его в общем модуле *ПечатьДокументов* (рис. 5.11).


```

Общий модуль ПечатьДокументов: Модуль
Процедура ПечатьНакладной(ТабДок, Ссылка) Экспорт
Макет = ПолучитьОбщийМакет("Накладная");
Запрос = Новый Запрос;
Запрос.Текст = ТекстЗапросаНакладная();
Запрос.Параметры.Вставить("Ссылка", Ссылка);
Выборка = Запрос.Выполнить().Выбрать();
СтрТаб = Макет.ПолучитьОбласть("СтрТаб");
Шапка = Макет.ПолучитьОбласть("Шапка");
Подвал = Макет.ПолучитьОбласть("Подвал");
ТабДок.Очистить();
ВставлятьРазделительСтраниц = Ложь;
Пока Выборка.Следующий() Цикл
    Если ВставлятьРазделительСтраниц Тогда
        ТабДок.ВывестиГоризонтальныйРазделительСтраниц();
    КонецЕсли;
    Шапка.Параметры.Заполнить(Выборка);
    Шапка.Параметры.ДатаДок = формат(Выборка.ДатаДок, "дд.мм.yyyy");
    ТабДок.Вывести(Шапка, Выборка.Уровень());
    ВыборкаТовары = Выборка.Товары.Выбрать();
    Пока ВыборкаТовары.Следующий() Цикл
        СтрТаб.Параметры.Заполнить(ВыборкаТовары);
        ТабДок.Вывести(СтрТаб, ВыборкаТовары.Уровень());
    КонецЦикла;
    Подвал.Параметры.СуммаДокумента = СтрШаблон("%1 руб. %2 коп.",
        Цел(Выборка.СуммаДокумента),
        формат(100 * (Выборка.СуммаДокумента - Цел(Выборка.СуммаДокумента)),
            "Цц=2; Цн=00; ЦВн="));
    ТабДок.Вывести(Подвал);
    ВставлятьРазделительСтраниц = Истина;
КонецЦикла;
КонецПроцедуры

```

Рис. 5.11. Код процедуры *ПечатьНакладной*

Необходимые для печати данные формируются следующим запросом.

```

"ВЫБРАТЬ
    Поступление.Номер КАК Номер,
    Поступление.Дата КАК ДатаДок,
    Поступление.Контрагент.Наименование КАК Поставщик,
    НаименованиеОрганизации.Значение КАК Покупатель,
    Поступление.СуммаДокумента КАК СуммаДокумента,
    Поступление.Товары.(
        НомерСтроки КАК НомерСтроки,
        Препарат КАК Препарат,
        Препарат.ЕдиницаИзмерения КАК ЕдИзм,
        Количество КАК Кол,
        Цена КАК Цена,
        Сумма КАК Сумма
    ) КАК Товары
ИЗ
    Документ.Поступление КАК Поступление,
    Константа.НаименованиеОрганизации КАК НаименованиеОрганизации
ГДЕ
    Поступление.Ссылка В(&Ссылка)
ОБЪЕДИНИТЬ ВСЕ
ВЫБРАТЬ
    Продажа.Номер,
    Продажа.Дата,
    НаименованиеОрганизации.Значение,
    "" _____ "",
    Продажа.СуммаДокумента,
    Продажа.Товары.(
        НомерСтроки,
        Препарат,
        Препарат.ЕдиницаИзмерения,
        Количество,
        Цена,
        Сумма
    )
ИЗ
    Документ.Продажа КАК Продажа,
    Константа.НаименованиеОрганизации КАК НаименованиеОрганизации
ГДЕ
    Продажа.Ссылка В(&Ссылка)";

```

Запрос состоит из двух объединенных запросов, выбирающих данные из таблиц различных документов, но приводящих их к единообразному виду.

Например, в первом запросе в поле «Поставщик» выводится наименование контрагента, а во втором – наименование нашей организации.

Для запуска формирования формы накладной пользователем создается общая команда *ПечатьНакладной* (рис. 5.12).

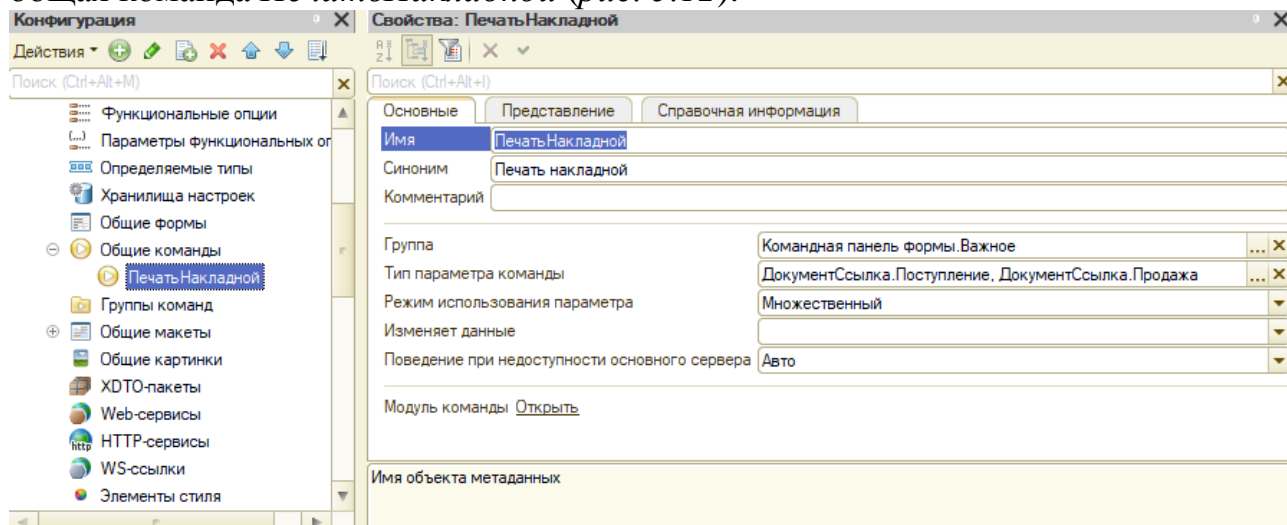


Рис. 5.12. Настройки общей команды

Код модуля команды приведен на рис. 5.13.

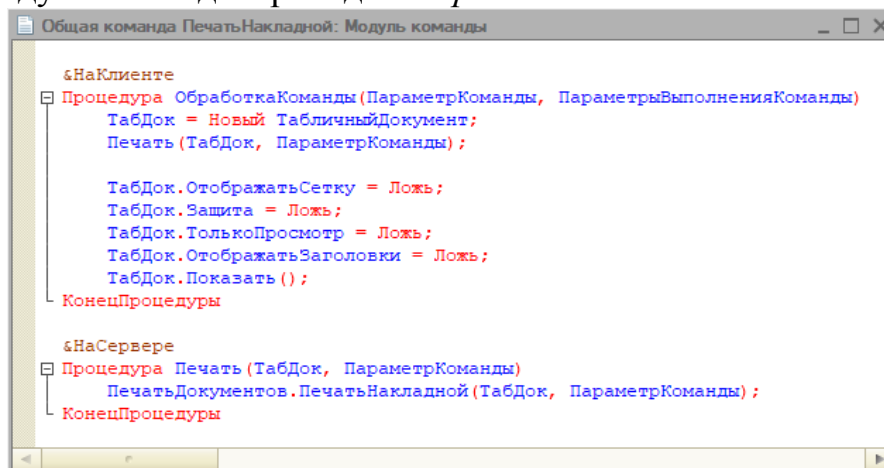


Рис. 5.13. Код модуля общей команды

Следует отметить, что в этом случае кнопка команды автоматически появляется в формах документах и их списках и может быть использована как для печати одного документа, так и нескольких.

Для демонстрации получения аналитической информации создадим отчет *АнализПродаж*. Отчет будет формироваться на основании данных регистра *ПродажиНоменклатуры*. Подробнее создание отчета с использованием СКД рассмотрено в текущем курсе, а также в разделе 4. Поэтому здесь приведем только текст запроса для набора данных.

ВЫБРАТЬ

ПродажиНоменклатурыОбороты.Номенклатура КАК Препарат,
 ПродажиНоменклатурыОбороты.КоличествоОборот КАК Количество,
 ПродажиНоменклатурыОбороты.СебестоимостьОборот КАК Себестоимость,
 ПродажиНоменклатурыОбороты.СтоимостьПродажиОборот КАК Выручка,
 ПродажиНоменклатурыОбороты.СтоимостьПродажиОборот - ПродажиНоменклатурыОбороты.СебестоимостьОборот

КАК Прибыль
 ИЗ

РегистрНакопления.ПродажиНоменклатуры.Обороты(, ,) КАК ПродажиНоменклатурыОбороты

«Автозаполнение» оставляем включенным, на закладке «Ресурсы» в ресурсы выносим все четыре числовые поля, на закладке «Параметры» автомати-

чески появляются *НачалоПериода* и *КонецПериода* (так как используется регистр в запросе), на закладке «Настройки» с помощью конструктора настроек устанавливаем группировку по *Препарат*, вывод всех ресурсов и упорядочивание по *Препарат*. Здесь же включаем параметры отчета в пользовательские настройки (см. раздел 4), настраиваем возможность отбора пользователем по полю *Номенклатура* (это измерение регистра из его виртуальной таблицы, отбор по нему предпочтительнее, чем по колонке *Препарат*, хотя в ней та же номенклатура). Можно вынести этот отбор в пользовательские настройки, чтобы он отображался, как и параметры, на форме отчета. Галочку у отбора лучше снять, иначе по умолчанию отчет сформируется у пользователя с отбором по пустой номенклатуре. Заголовок поля отбора можно переименовать в «Препарат» здесь же («Установить представление» в контекстном меню) либо на закладке «Наборы данных» («Заголовок» у поля *Номенклатура*). Чтобы заголовок таблицы отчета выводился полужирным шрифтом, на закладке «Условное оформление» добавим оформление *Шрифт* (установим его полужирным), в качестве оформляемых полей выберем колонки отчета, в «Область использования» выберем *в заголовке полей*. Чтобы в печатную форму над таблицей выводился заголовок отчета, на закладке «Другие настройки» для настройки «Заголовки» напишем «Анализ продаж». Приветствуются при разработке отчетов СКД самостоятельность и творческий подход к настройкам. Настроек много и их применение лучше осваивается на практике при самостоятельном использовании.

По условию задачи необходимо реализовать рабочее место провизора. Сделаем его в виде обработки *РабочееМестоПровизора* (рис. 5.14).

Рис. 5.14. Форма обработки *РабочееМестоПровизора*

В верхней части формы обработки пользователь вводит необходимые значения отбора. При изменении значения любого из трех полей отбора запросом отбираются препараты, соответствующие значениям заполненных полей отбора. Это не совсем верное решение с практической точки зрения в реальных информационных базах с тысячами позиций номенклатуры, но в небольшой учебной базе вполне допустимо. Отобранные запросом значения попадают в

таблицу в средней части формы. Двойным кликом мыши на строке в этой таблице провизор переносит выбранный препарат в таблицу в нижней части формы, которая соответствует табличной части Товары документа Продажа. Провизор может изменить количество отпускаемого препарата, но не может поменять цену и сумму. При нажатии на кнопку «Оформить продажу» автоматически создается и проводится документ *Продажа*, табличная часть которого заполняется строками из нижней таблицы формы. Если проведение документа прошло успешно, то нижняя таблица очищается, в противном случае выводится сообщение об ошибке, отобранные товары остаются в нижней таблице.

Нижняя таблица формы реализована в виде табличной части обработки, чтобы иметь возможность вывести на форму итог по сумме, так как в этом случае не нужно писать никакого дополнительного кода. Можно просто скопировать табличную часть *Товары* из документа *Продажа* и вставить её в обработку.

Средняя таблица реализуется с помощью реквизита формы *ОтобранныеПрепараты* (тип *ТаблицаЗначений*). Для отбора используются реквизиты формы *НаименованиеПрепарата*, *Назначение* и *ДействующееВещество* соответствующих типов, которые размещаются в горизонтальной группе в верхней части формы. К элементам формы, связанным с этими реквизитами, подключаются обработчики событий *ПриИзменении* (рис. 5.15).

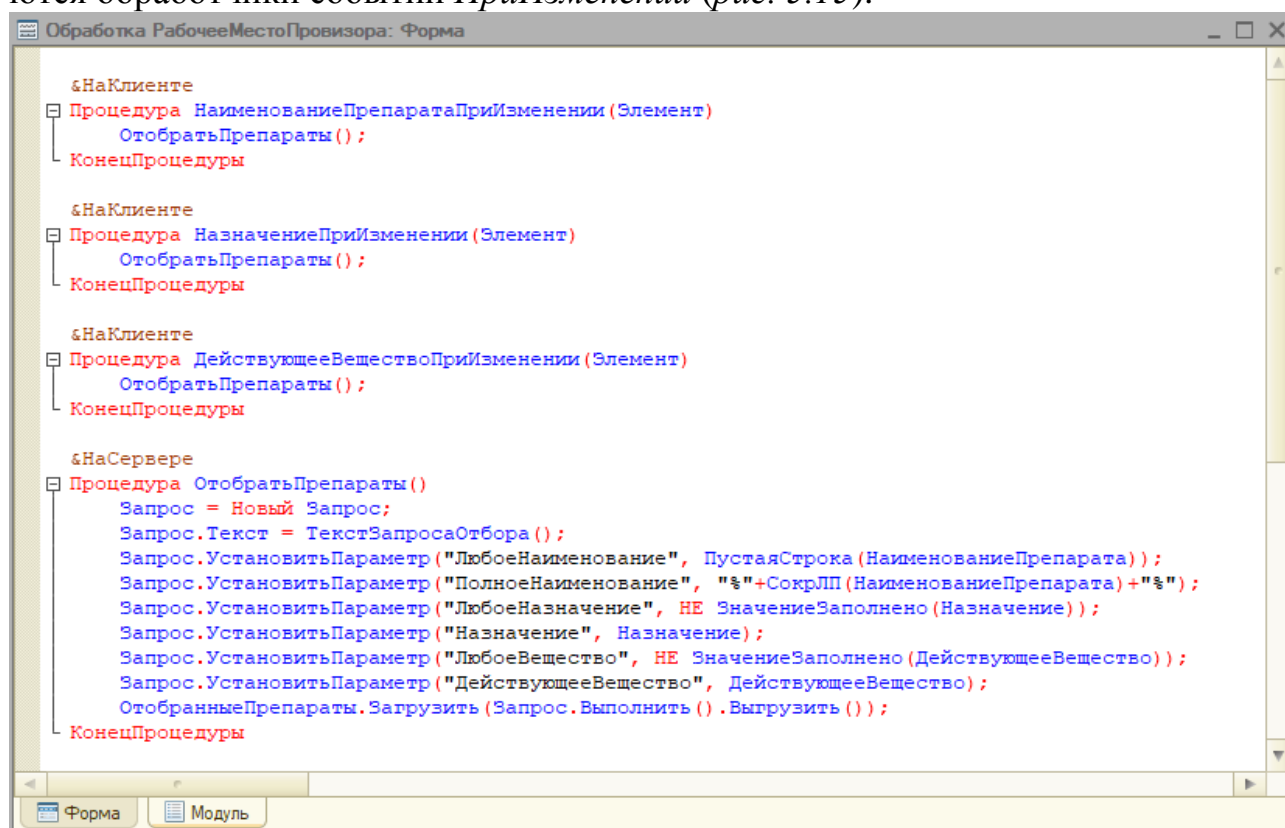


Рис. 5.15. Обработчики изменения значений отборов

При изменении любого из полей отбора выполняется запрос, заполняющий таблицу *ОтобранныеПрепараты*. Ниже приводится текст этого запроса.

```

"ВЫБРАТЬ
|         СписокНоменклатуры.Ссылка КАК Ссылка
|ПОМЕСТИТЬ ВтОтобранныеПрепараты
|ИЗ
|         Справочник.Номенклатура КАК СписокНоменклатуры
  
```

```

ГДЕ
    НЕ СписокНоменклатуры.ЭтоГруппа
    И НЕ СписокНоменклатуры.ПометкаУдаления
    И (&ЛюбоеНаименование
        ИЛИ СписокНоменклатуры.ПолноеНаименование ПОДОБНО &ПолноеНаименование)
    И (&ЛюбоеНазначение
        ИЛИ СписокНоменклатуры.Назначение = &Назначение)
    И (&ЛюбоеВещество
        ИЛИ СписокНоменклатуры.ДействующееВещество = &ДействующееВещество)
;

////////////////////////////////////
ВЫБРАТЬ
    ОстаткиНоменклатуры.Остатки.Номенклатура КАК Номенклатура,
    ОстаткиНоменклатуры.Остатки.КоличествоОстаток КАК КоличествоОстаток
ПОМЕСТИТЬ ВтОстатки
ИЗ
    РегистрНакопления.ОстаткиНоменклатуры.Остатки(
        Номенклатура В
        (ВЫБРАТЬ
            Вт.Ссылка
        ИЗ
            ВтОтобранныеПрепараты КАК Вт)) КАК ОстаткиНоменклатуры.Остатки
;

////////////////////////////////////
ВЫБРАТЬ
    ЦеныНоменклатурыСрезПоследних.Номенклатура КАК Номенклатура,
    ЦеныНоменклатурыСрезПоследних.Цена КАК Цена
ПОМЕСТИТЬ ВтЦены
ИЗ
    РегистрСведений.ЦеныНоменклатуры.СрезПоследних(
        Номенклатура В
        (ВЫБРАТЬ
            Вт.Ссылка
        ИЗ
            ВтОтобранныеПрепараты КАК Вт)) КАК ЦеныНоменклатурыСрезПоследних
;

////////////////////////////////////
ВЫБРАТЬ
    ВтОтобранныеПрепараты.Ссылка КАК Препарат,
    ВтОстатки.КоличествоОстаток КАК Остаток,
    ВтЦены.Цена КАК Стоимость
ИЗ
    ВтОтобранныеПрепараты КАК ВтОтобранныеПрепараты
    ЛЕВОЕ СОЕДИНЕНИЕ ВтОстатки КАК ВтОстатки
    ПО ВтОтобранныеПрепараты.Ссылка = ВтОстатки.Номенклатура
    ЛЕВОЕ СОЕДИНЕНИЕ ВтЦены КАК ВтЦены
    ПО ВтОтобранныеПрепараты.Ссылка = ВтЦены.Номенклатура
УПОРЯДОЧИТЬ ПО
    Ссылка";

```

Использование временных таблиц существенно ускоряет выполнение подобных запросов в больших информационных базах.

Для элемента формы *ОтобранныеТовары* установлено свойство *ТолькоПросмотр*, что позволяет задействовать механизм выбора строк двойным щелчком мыши. При этом происходит событие *Выбор* (рис. 5.16).

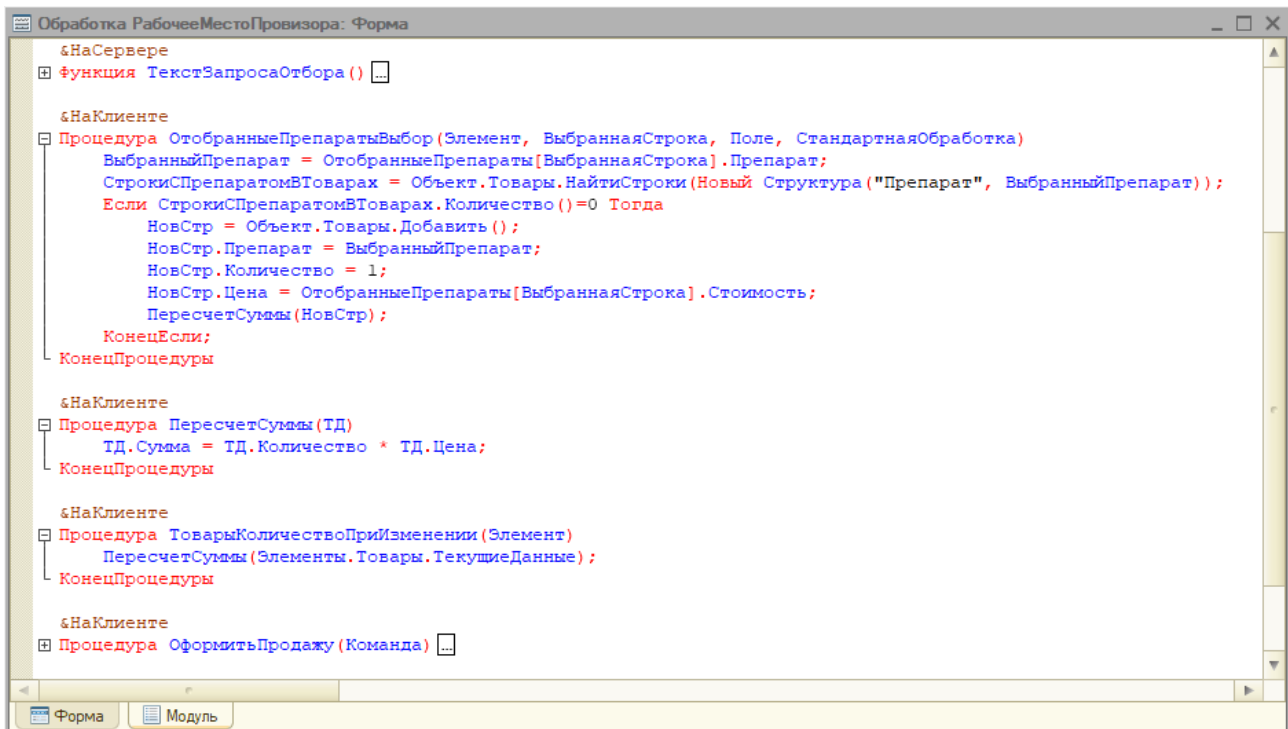


Рис. 5.16. Обработка выбора препарата в таблице *ОтобранныеПрепараты*

Сначала происходит проверка, присутствует ли выбранный препарат в таблице *Товары*. Если нет, то в неё добавляется новая строка.

Процедура *ПересчетСуммы* используется и для расчета суммы при подборе товара, и при изменении количества в таблице *Товары*. В этой таблице установлено свойство *ТолькоПросмотр* для всех колонок, кроме *Количество*.

Нажатие на кнопку «Оформить продажу» запускает действие соответствующей команды (рис. 5.17).

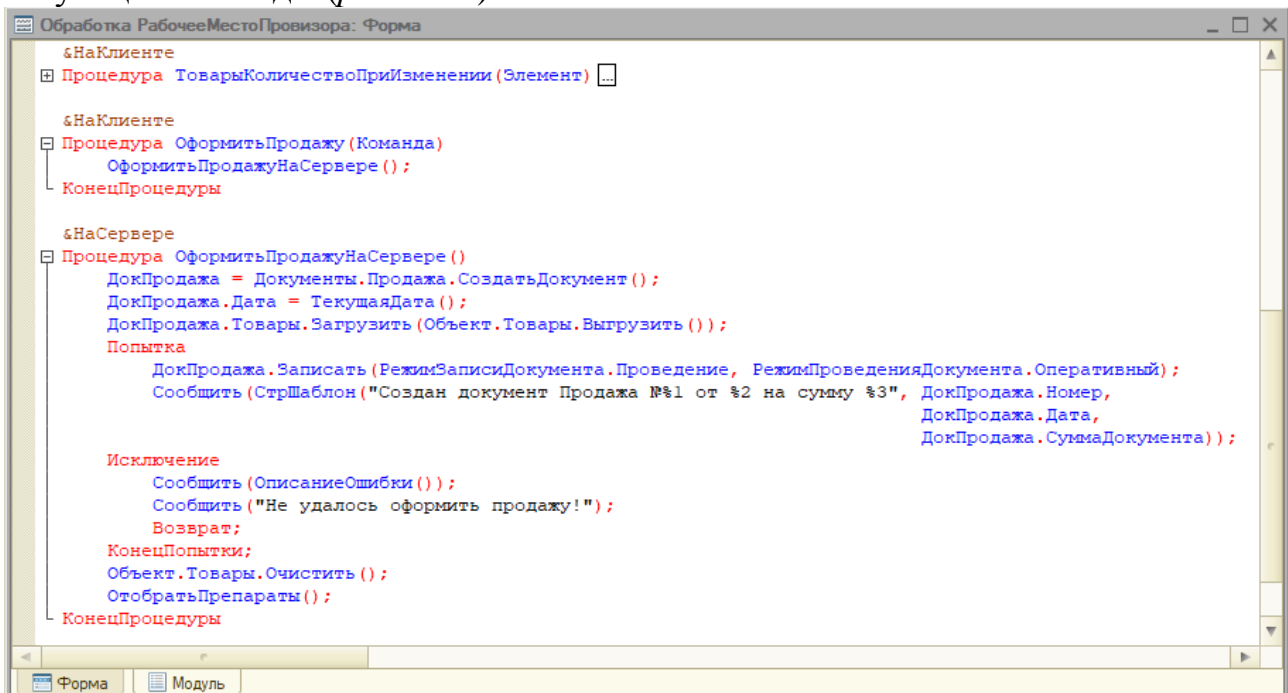


Рис. 5.17. Создание документа *Продажа*

Создается новый документ *Продажа*, заполняется и проводится в оперативном режиме в попытке. Если по какой-либо причине проведение оказывается невозможным (например, недостаточно товара в аптеке), то возникает ис-

ключение, и пользователь получает соответствующее сообщение о невозможности оформить продажу. В конце вызывается процедура *ОтобразитьПрепараты* для обновления значения остатка товара в таблице *ОтобранныеПрепараты* после продажи.

Данный вариант реализации рабочего места – не единственный. Возможно решение с динамическим списком справочника, с кэширование отборов и прочее. Встроенный язык 1С дает богатые возможности для программиста реализовать себя.

5.4. Создание ролей

Обычно созданной информационной базой пользуются несколько сотрудников, поэтому для них необходимо разграничение функциональности системы в зависимости от их должностных обязанностей либо, как минимум, регистрация наиболее важных действий, совершаемых ими в системе.

С данной системой будут работать минимум два пользователя – Администратор, имеющий полные права, и Провизор, в обязанности которого входит только оформлять продажи с помощью своего рабочего места и просматривать отчет «Анализ продаж».

Для начала создадим две подсистемы как для организации работы с объектами метаданных в режиме «1С:Предприятие», так и для упорядочивания объектов конфигурации в режиме разработки. Заметим, что в практической работе создание новой системы либо модернизацию существующей начинают именно с создания структуры подсистем и новые объекты сразу связывают с одной из них.

В данной учебной конфигурации создадим две подсистемы: *НСИиАдминистрирование* и *Провизор*. В подсистему *Провизор* войдут только два объекта: обработка *РабочееМестоПровизора* и отчет *АнализПродаж*.

Все остальные объекты конфигурации войдут в состав *НСИиАдминистрирование*, причем в учебных целях мы создадим в ней две подчиненные подсистемы – *Справочники* и *Документы* – и разместим в них соответствующие объекты. После распределения объектов по подсистемам рекомендуется проверить, не осталось ли объектов, не входящих ни в одну подсистему (рис. 5.18).

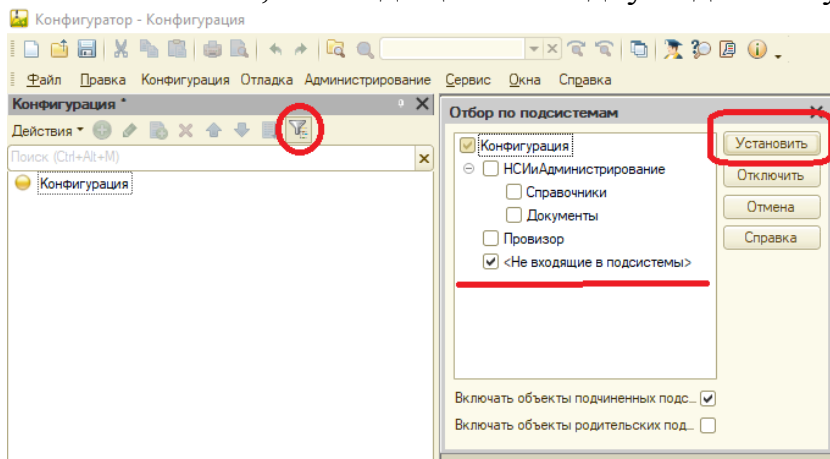


Рис. 5.18. Отбор объектов конфигурации по подсистемам

Как вариант, можно следующим образом (хотя и не обязательно) настроить окно клиентского приложения (рис. 5.19). Для этого надо сначала правой кнопкой мыши открыть контекстное меню на корне конфигурации и выбрать «Открыть интерфейс клиентского приложения». Затем мышью перетащить панели в левой части нужным образом.

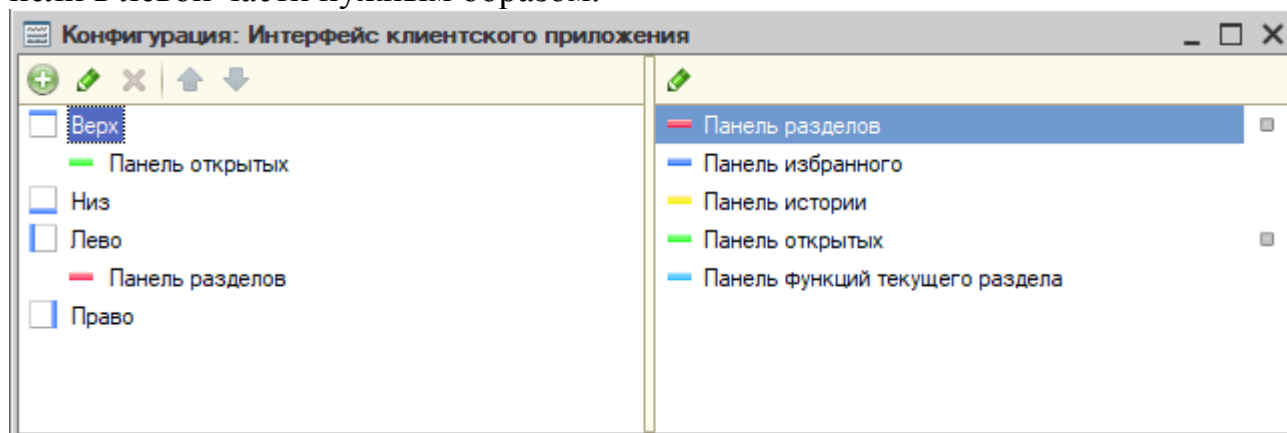


Рис. 5.19. Вариант настройки интерфейса

Теперь необходимо разграничить права *Администратора* и *Провизора*. В современных типовых конфигурациях для организации доступа и управления пользователями используются комплексные механизмы, объединяющие как объекты конфигурации, так и возможности платформы по управлению пользователями. Эти механизмы включают в себя множество ролей, каждая из которых определяет права доступа только к одному из объектов метаданных (документу, справочнику и так далее). При этом создаются отдельно роли на чтение и на добавление/изменение данных в этих объектах. Все пользователи учитываются в справочнике *Пользователи*, связанном со списком пользователей, который ведет платформа. Для наделения пользователя необходимыми элементарными правами, заданными в ролях, используется справочник *ПрофилиГруппДоступа*, который хранит целый набор ролей, подключаемых пользователю при его регистрации в системе в момент начала работы.

В данной учебной задаче мы упростим управление пользователями и их правами. Для этого создадим всего две роли: *ПолныеПрава* и *Провизор*. В этих ролях зададим все необходимые разрешения, а затем создадим и подключим эти роли двум тестовым пользователям, которых также создадим в конфигураторе.

В роли *ПолныеПрава* выставляем почти все галочки. Снять их только рекомендуется:

- у всех справочников: «Интерактивное удаление», «Интерактивное удаление предопределенных», «Интерактивная пометка на удаление предопределенных», «Интерактивное снятие пометки на удаление предопределенных» и «Интерактивное удаление помеченных предопределенных»;

- всех документов: «Интерактивное удаление».

Это делается для того, чтобы пользователь с полными правами не мог по ошибке нарушить целостность информационной базы.

В нижней части окна настройки роли *ПолныеПрава* рекомендуется выставить галочки «Устанавливать права для новых объектов» и «Устанавливать права для реквизитов и табличных частей по умолчанию».

В роли *Провизор* необходимо выставить следующие галочки:

- *Конфигурация*: «Тонкий клиент», «Веб-клиент», «Мобильный клиент», все начинающиеся с «Режим основного окна...», «Сохранение данных пользователя»;

- подсистема *Провизор*: «Просмотр»;

- общие команды *ПечатьНакладной*: «Просмотр»;

- константа *НаименованиеОрганизации*: «Чтение»;

- все справочники: «Чтение», «Просмотр», «Ввод по строке»;

- документ *Поступление*: «Чтение», «Просмотр», «Ввод по строке»;

- документ *Продажа*: «Чтение», «Добавление», «Изменение», «Проведение», «Отмена проведения», «Просмотр», «Редактирование», «Интерактивная пометка на удаление», «Интерактивное снятие пометки удаления», «Интерактивное проведение», «Интерактивное проведение неоперативное», «Интерактивная отмена проведения», «Интерактивное изменение проведенных», «ввод по строке»;

- отчет *АнализПродаж*: «Использование», «Просмотр»;

- обработка *РабочееМестоПровизора*: «Использование», «Просмотр»;

- регистр сведений *ЦеныНоменклатуры*: «Чтение»;

- регистры накопления: «Чтение».

Теперь в конфигураторе в «Администрирование» - «Пользователи» создаем двух пользователей (Администратор и Провизор) и назначаем им соответствующие роли. Проверяем работоспособность системы под каждым из этих пользователей.

6. КОНТРОЛЬНЫЕ ЗАДАНИЯ

6.1. Учет товаров в разрезе складов

Необходимо разработать конфигурацию для учета товаров. Суммовой учет не ведется. Взаиморасчеты с покупателями и поставщиками не ведутся. Учет товаров ведется в разрезе складов. В системе необходимо регистрировать два вида операций: Поступление товара, Продажу товара.

При поступлении товара Пользователь в табличной части указывает какие товары и в каком количестве поступили в организацию. Необходимо предусмотреть учет до граммов. В шапке документа выбирается склад. При продаже товаров указывается какие товары были проданы и в каком количестве с какого склада. Склад выбирается для каждого товара в табличной части. Продать товар "в минус" нельзя. То есть в момент продажи необходимо проверять остаток товара.

Необходимо построить отчет по остаткам товаров следующего вида (табл. 6.1).

Т а б л и ц а 6.1. Остатки товаров на ... (заданную дату)

Товар/Склад	Юг	Север	Запад	Итого
Ложка	100.000	40.000		140.000
Вилка	45.000		80.000	125.000
Поварешка		12.000	1.000	13.000

Отчет строиться на конец дня, указанного пользователем. Особое внимание следует уделить последней секунде дня. Документы, записанные на эту секунду, должны попадать в отчет.

6.2. Учет себестоимости в пределах компании

Необходимо разработать конфигурацию для учета товаров. Многоскладской учет не ведется. Взаиморасчеты с покупателями и поставщиками не ведутся. В системе необходимо регистрировать два вида операций: Поступление товара, Продажу товара.

При поступлении товара Пользователь в табличной части указывает какие товары и в каком количестве поступили в организацию. Также необходимо указывать закупочную цену товаров. Необходимо предусмотреть учет до граммов. При продаже товаров указывается какие товары были проданы и в каком количестве, на какую сумму. При продаже товара необходимо рассчитать стоимость списываемых товаров. Расчет стоимости необходимо произвести по методу "Средневзвешенная-скользящая". То есть на момент продажи необходимо определить суммовой и количественный остаток и рассчитать сумму к списанию

$$\text{СуммаСписания} = \text{ОстатокСуммы} / \text{ОстатокКоличества} * \text{СколькоСписать}$$

Продать товар "в минус" нельзя. То есть в момент продажи необходимо проверять остаток товара. Важно помнить, что пользователь может вводить документы задним числом! Необходимо построить отчет по прибыли товаров следующего вида (табл. 6.2).

Т а б л и ц а 6.2. Продажи товаров за ... (с даты начала по дату окончания заданного периода)

Товар	Себестоимость	Выручка	Прибыль
Ложка	100.00	400.00	300.00
Вилка	148.00	950.30	802.30
Поварешка	2.00	800.00	798.00

6.3. Учет себестоимости в разрезе складов

Необходимо разработать конфигурацию для учета товаров. Ведется учет в разрезе складов. Взаиморасчеты с покупателями и поставщиками не ведутся. В системе необходимо регистрировать два вида операций: Поступление товара, Продажу товара.

При поступлении товара Пользователь в табличной части указывает какие товары и в каком количестве поступили в организацию, а в шапке документ указывается склад поступления. Также необходимо указывать закупочную цену товаров. Необходимо предусмотреть учет до граммов. При продаже товаров указывается какие товары были проданы и в каком количестве, на какую сумму, в шапке документа указывается склад списания. При продаже товара необходимо рассчитать стоимость списываемых товаров. Расчет стоимости необходимо произвести по методу "Средневзвешенная-скользящая". То есть на момент продажи необходимо определить суммовой и количественный остаток и рассчитать сумму к списанию.

$$\text{СуммаСписания} = \text{ОстатокСуммы} / \text{ОстатокКоличества} * \text{СколькоСписать}$$

Себестоимость рассчитывается по всем складам. То есть если на склад "Юг" поступила одна Ложка, и закупочная цена у нее была 100, и такая-же Ложка поступила на склад "Север" по 200, то при списании Ложки с любого склада себестоимость списания должна быть 150.

Продать товар "в минус" нельзя. То есть в момент продажи необходимо проверять остаток товара. Важно помнить, что пользователь может вводить документы задним числом! Необходимо построить отчет по прибыли товаров следующего вида (табл. 6.2).

6.4. Контроль себестоимости остатков товаров

Необходимо разработать конфигурацию для учета товаров. Многоскладской учет не ведется. Взаиморасчеты с покупателями и поставщиками не ведутся. В системе необходимо регистрировать два вида операций: Поступление товара, Продажу товара.

При поступлении товара Пользователь в табличной части указывает какие товары и в каком количестве поступили в организацию. Также необходимо указывать закупочную цену товаров. Необходимо предусмотреть учет до граммов. При продаже товаров указывается какие товары были проданы и в каком количестве. При продаже товара необходимо рассчитать стоимость списываемых товаров. Расчет стоимости необходимо произвести по методу "Средневзвешенная-скользящая". То есть на момент продажи необходимо определить суммовой и количественный остаток и рассчитать сумму к списанию.

$$\text{СуммаСписания} = \text{ОстатокСуммы} / \text{ОстатокКоличества} * \text{СколькоСписать}$$

Продать товар "в минус" нельзя. То есть в момент продажи необходимо проверять остаток товара. Важно помнить, что пользователь может вводить документы задним числом! Необходимо построить отчет по остаткам товаров следующего вида (табл. 6.3).

Таблица 6.3. Остатки товаров на ... (заданную дату)

Товар	Остаток	Стоимость
Ложка	100.000	400.00
Вилка	148.000	950.30
Поварешка	2.000	800.00

Отчет строиться на конец дня, указанного пользователем. Особое внимание следует уделить последней секунде дня. Документы, записанные на эту секунду, должны попадать в отчет.

6.5. Расчет прибыли при списании по методу FIFO

Необходимо разработать конфигурацию для учета товаров. Многоскладской учет не ведется. Взаиморасчеты с покупателями и поставщиками не ведутся. В системе необходимо регистрировать два вида операций: Поступление товара, Продажу товара. При поступлении товара Пользователь в табличной части указывает какие товары и в каком количестве поступили в организацию. Также необходимо указывать закупочную цену товаров. Необходимо предусмотреть учет до граммов. При продаже товаров указывается какие товары были проданы и в каком количестве, на какую сумму.

При продаже товара необходимо рассчитать стоимость списываемых товаров. Расчет стоимости необходимо произвести по методу "FIFO" (First-In-First-Out). То есть на момент продажи необходимо определить какие партии остались и списывать себестоимость с учетом хронологии их поступления. К примеру, товар "ложка" поступал трижды (тремя партиями) и стоил 10, 20, 30 рублей за штуку. При списании двух ложек необходимо списать первые две партии на сумму 10 и 20 рублей. Если партия списывается частично, то сумма списания (внутри одной партии) рассчитывается как средняя:

$$\text{СуммаСписания} = \text{ОстатокСуммы} / \text{ОстатокКоличества} * \text{СколькоСписать}$$

Продать товар "в минус" нельзя. То есть в момент продажи необходимо проверять остаток товара. Важно помнить, что пользователь может вводить документы задним числом! Необходимо построить отчет по прибыли товаров следующего вида (табл. 6.2).

6.6. Учет при списании сроков годности

Необходимо разработать конфигурацию для учета товаров. Многоскладской учет не ведется. Взаиморасчеты с покупателями и поставщиками не ведутся. В системе необходимо регистрировать два вида операций: Поступление товара, Продажу товара. В системе необходимо регистрировать два вида операций: Поступление товара, Продажу товара.

При поступлении товара Пользователь в табличной части указывает какие товары и в каком количестве поступили в организацию. Также необходимо указывать закупочную цену товаров. Необходимо предусмотреть учет до граммов. При поступлении товаров указывается срок годности партии, для каждого

товара свой. При продаже товаров указывается какие товары были проданы и в каком количестве, на какую сумму.

При продаже товара необходимо списывать те товары, срок годности которых подходит к концу. К примеру, если поставка молока "Буренка" поступала со сроком годности 30.10.2020 и 31.01.2020, то сначала списывается партия со сроком годности 30.01.2020. Если партия списывается частично, то расчет себестоимости списания происходит по среднему (внутри партии)

$$\text{СуммаСписания} = \text{ОстатокСуммы} / \text{ОстатокКоличества} * \text{СколькоСписать}$$

Продать товар "в минус" нельзя. То есть в момент продажи необходимо проверять остаток товара. Важно помнить, что пользователь может вводить документы задним числом! Необходимо построить отчет по прибыли товаров следующего вида (табл. 6.4).

Т а б л и ц а 6 . 4 . Остатки товаров на ... (заданную дату)

Товар	Срок годности	Количество	Стоимость
Молоко "Буренка"	28.01.2020	10	100
Молоко "Буренка"	30.01.2020	15	200
Молоко "Буренка"	31.01.2020	5	300

6.7. Указание конкретных сроков годности при списании товаров

Необходимо разработать конфигурацию для учета товаров. Многоскладской учет не ведется. Взаиморасчеты с покупателями и поставщиками не ведутся. В системе необходимо регистрировать два вида операций: Поступление товара, Продажу товара. В системе необходимо регистрировать два вида операций: Поступление товара, Продажу товара.

При поступлении товара Пользователь в табличной части указывает какие товары и в каком количестве поступили в организацию. Также необходимо указывать закупочную цену товаров. Необходимо предусмотреть учет до граммов. При поступлении товаров указывается срок годности партии, для каждого товара свой. При продаже товаров указывается какие товары были проданы и в каком количестве, на какую сумму. При продаже товара, при заполнении табличной части необходимо предоставить возможность пользователю выбирать конкретную партию списания (срок годности). Продать товар "в минус" нельзя. То есть в момент продажи необходимо проверять остаток товара по конкретной партии (сроку годности). Важно помнить, что пользователь может вводить документы задним числом!

Необходимо построить отчет по прибыли товаров следующего вида (табл. 6.4).

6.8. Резервирование товаров

Необходимо разработать конфигурацию для учета товаров. Многоскладской учет не ведется. Взаиморасчеты с покупателями и поставщиками не ведутся. В системе необходимо регистрировать три вида операций: Поступление товара, Продажу товара. Резервирование товаров.

Поступление товаров производится документами Приходная накладная. В документе в табличной части указывается какие товары и в каком количестве поступили.

Покупатели по телефону могут зарезервировать товары. В этом случае в систему вводят документ "Резервирование товара". В документе указывается покупатель, для которого резервируют товары, сами товары и количество. Зарезервировать товары которых нет в наличии нельзя.

При продаже товара необходимо проверять есть ли свободные товары в наличии, в том случае если товара не хватает, необходимо уведомить пользователя о свободном остатке и не позволять проводить документ.

В том случае если покупатель забирает товары, которые он заранее зарезервировал - резерв должен автоматически уменьшится.

Необходимо разработать отчет (табл. 6.5).

Т а б л и ц а 6 . 5 . Остатки товаров на ... (заданную дату)

Товар	Остаток	В резерве	Свободный остаток
Ложка	10	1	9
Вилка	15	3	12

6.9. Расчеты с покупателями

Необходимо разработать конфигурацию для учета расчетов с покупателями. Расчеты с покупателями ведутся как в рублях, так и в иностранной валюте. Причем с одним покупателем взаиморасчеты могут вестись сразу в нескольких валютах.

При вводе документа "Акт об оказании услуг" регистрируется долг покупателя за оказанные услуги. В документе пользователь выбирает покупателя, валюту взаиморасчетов и сумму. Погашается долг покупателя документом "Приходный кассовый ордер". В документе пользователь выбирает "Акт", долг по которому погашается. Долг может быть погашен полностью или частично. Считается, что авансы покупателя не платят и переплат не делают, однако это контролировать не нужно. Долг может быть погашен как в валюте, в которой он был образован, так и в рублях, по курсу валюты на дату оплаты. К примеру, Акт был выписан на \$100, а Приходный кассовый ордер вводится в рублях.

В любой момент необходимо построить отчет "Дебиторская задолженность" (табл. 6.6).

Т а б л и ц а 6.6. Список дебиторов на ... (заданную дату)

Покупатель	Валюта	Сумма в валюте	Сумма рублевого покрытия
Мир	USD	100	6 298,20
Мир	EUR	150	10 601,78
Свет	Руб.	7500	7500

6.10. Дни рождения студентов

Студенты группы №42 очень сплоченный и дружный коллектив. Они дарят на день рождения друг друга подарки, предварительно "скинувшись" всей группой.

Деньги на подарки собирают раз в месяц в первую неделю. А подарки покупают и дарят либо в сам день рождения, либо заранее, если день рождения выпадает на праздники или выходные.

Следует разработать базу данных для максимальной автоматизации этой задачи. А именно:

1. Создать список студентов, с указанием ФИО, дата рождения и пол (девочкам всегда покупают не только подарок, но и цветы!)
2. Создать документ, в котором можно указать кто и сколько сдал на подарок, для кого предназначается сбор.
3. Создать документ "Выдача денег на подарок", в документе нужно указать кому и сколько выдано и для кого планируется покупать подарок.
4. После покупки подарка пользователь вводит еще один документ "Подарок прибыл", в котором указывают за сколько фактически купили подарок.
5. Факт вручения подарка тоже вводят в базу документом.

Раз в месяц, ответственный за ведение учета в базе формирует отчеты со списком тех, у кого день рождения в этом месяце (табл. 6.7), какого числа их поздравлять и сколько денег собрано (табл. 6.8).

Т а б л и ц а 6.7. Отчет дням рождениям за ... (с даты начала по дату окончания заданного периода)

ФИО	День рождения	Стукнет лет
Иванюхин	12.01.2000	20
Каменский	18.01.1998	22
Швецов	31.01.1999	21
Чистов	31.01.2001	19

Таблица 6.8. Предстоящие поздравления за ... (с даты начала по дату окончания заданного периода)

ФИО	День рождения	Когда поздравляем	Сколько собрано
Иванюхин	12.01.2000	10.01.2020	8500
Каменский	18.01.1998	17.01.2020	7300
Швецов	31.01.1999	31.01.2020	2900

Если дни рождения выпадают на выходные, то поздравлять их надо накануне в будние дни.

ЗАКЛЮЧЕНИЕ

Рассмотренные в методических указаниях практические задачи разработки программных приложений на платформе «1С:Предприятие 8» развивают у обучающегося способности использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности; решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности; разрабатывать алгоритмы и программы, пригодные для практического применения.

При решении задач с использованием платформы «1С:Предприятие 8» особенно важна их практическая направленность на автоматизацию учета самых разнообразных, подчас рутинных видов деятельности. При этом минимизируются затраты на разработку, выражающиеся в первую очередь в человеко-часах труда высококвалифицированных программистов.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Классификация языка программирования «1С:Язык программирования»: базовые свойства, признаки, формат языковых конструкций.
2. Классификация языка программирования «1С:Язык программирования»: синтаксис и семантика.
3. Среда разработки платформы.
4. Понятие контекста исполнения. Модули. Структура модулей.
5. Типы данных. Базовые операторы.
6. Арифметические, логические операции, условные операторы, циклы.
7. Процедуры и функции.
8. Массивы, описание и методы объекта.
9. Списки значений, описание и методы объекта.
10. Таблица значений, описание и методы объекта.
11. Работа с файловой системой.
12. Диалоговые функции.
13. Библиотека встроенных функций: обработка числовых типов, строковых, дат.
14. Преобразование типов данных.
15. Разворачивание пустой конфигурации. Совместная работа над конфигурацией.
16. Создание констант, справочников, документов, регистров.
17. Создание подсистем и ролей.
18. Разработка программного кода. Размещение его в программных модулях.
19. Формы. Обработчики событий.
20. Объект «Отчеты». Разработка аналитических отчетов.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. *Тагайцева С. Г., Юрченко Т. В.* Предметно-ориентированное программирование. – Нижний Новгород: Нижегородский государственный архитектурно-строительный университет, ЭБС АСВ, 2018. – 89 с.
2. *Бойко Э. В.* 1С Предприятие 8.0. Универсальный самоучитель. – Саратов: Ай Пи Эр Медиа, 2010. – 375 с.
3. *Арсеньтьева А. Е.* 1С Предприятие. Шаг за шагом. Практическое пособие. – Саратов: Ай Пи Эр Медиа, 2009. – 217 с.
4. *Радченко М.Г., Хрусталева Е.Ю.* 1С:Предприятие 8.3. Практическое пособие разработчика. – М.: ООО «Публишинг», 2013. – 964 с.
5. *Хрусталева Е.Ю.* Разработка сложных отчетов в 1С:Предприятии 8. Система компоновки данных. – М.: ООО «Публишинг», 2008. – 513 с.