Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

АВТОМАТИЗАЦИЯ ЭКОНОМИЧЕСКОЙ И ОРГАНИЗАЦИОННОЙ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ

Методические указания к выполнению лабораторных работ
и организации самостоятельной работы для студентов направления
«Бизнес-информатика» (уровень бакалавриата)

2022

Сидоров Анатолий Анатольевич, Синчинова Людмила Иосифовна
Автоматизация экономической и организационной деятельности предприятия: Методические указания к выполнению лабораторных работ и организации самостоятельной работы для студентов направления «Бизнес-информатика» (уровень бакалавриата) / А.А. Сидоров, Л.И. Синчинова. – Томск, 2022. – 23 с.

Оглавление

1 Введение 4
2 Методические указания к проведению практических занятий 5
2.1 Практическое занятие «Установка системы 1С:Предприятие 8.3. Основные принципы работы с платформой» 5
Выводы 8
2.2 Практическое занятие «Разработка информационной системы для хранения информации о сотрудниках предприятия» 8
2.3 Практическое занятие «Разработка основных документов, отчеты» 11
2.4 Практическое занятие «Разработка запросов» 17
3 Методические указания для организации самостоятельной работы 22
3.1 Общие положения 22
3.2 Проработка лекционного материала 22
3.3 Подготовка к лабораторным занятиям 23
4. Рекомендуемая литература 23

1 Введение

Целью лабораторных занятий и самостоятельной работы по дисциплине «Автоматизация экономической и организационной деятельности предприятия» является закрепление теоретических основ базовых разделов экономической и организационной деятельности предприятия, а также формирование у студентов способности к самостоятельному или при помощи преподавателя анализу теоретического материала.

В результате проведения лабораторных занятий и самостоятельной работы студенты должны: получить знания об основных этапах установки и конфигурирования программы 1С:Предприятие, способах и правилах создания справочников для работы в этой программе, принципах и особенностях построения запросов.

2 Методические указания к проведению практических занятий

2.1 Практическое занятие «Установка системы 1С:Предприятие 8.3. Основные принципы работы с платформой»

Система 1С:Предприятие может работать в двух режимах. Первый называется "1С:Предприятие", второй - " *Конфигуратор* ". Разработка *прикладных решений* ведется в *конфигураторе*, а их *исполнение* - то есть - работа пользователей с ними - в режиме 1С:Предприятие.

Говоря о системе программ "1С:Предприятие" следует помнить, что существуют понятия " *платформа* " и " *конфигурация* ". ***Платформа*** - это среда, в которой разрабатывают и исполняют *конфигурации*. А *конфигурацию* можно сравнить с набором команд, для исполнения которых нужна *платформа*.

При запуске программы первым делом на экране появляется окно **Запуск 1С:Предприятия**. В нем нужно, во-первых, указать нужную *конфигурацию*, во-вторых - выбрать режим ее запуска. Если в списке пока нет *информационных баз* - *запуск* программы будет сопровождаться сообщением о том, что *список информационных баз* пуст и предложением добавить в него новую базу.

В области **Информационные базы** находится *список* подключенных *информационных баз*. В данный момент этот *список* пуст.

Окно содержит следующие кнопки:

- **1С:Предприятие**. Запуск системы в режиме 1С:Предприятие.
- **Конфигуратор**. Запуск системы в режиме *Конфигуратор*.
- **Добавить**. Запуск процесса добавления в список новой *информационной базы*.
- **Изменить**. Открывает окно изменения параметров добавленной *информационной базы*.
- **Удалить**. Удаляет из списка *информационную базу*.
- **Настройка**. Позволяет настроить внешний вид списка **Информационные базы**, установить каталог для поиска шаблонов *конфигураций* и обновлений.

Нажмем на кнопку **Добавить** (или ответим **Да** на вопрос о создании новой базы). Появится окно **Добавление информационной базы/группы**. Фактически, это мастер, который проводит вас через несколько шагов по добавлению базы в *список*

Здесь мы можем пойти двумя путями:

- Создание новой *информационной базы*.
- Добавление в список существующей *информационной базы*.

Нас интересует именно первый *пункт*, так как мы должны будем создать базу для последующей разработки в ней учебной *конфигурации*. Выберем его и нажмем на кнопку **Далее**. Появится окно, где можно выбрать вариант создания новой *информационной базы*.

Если ранее вы устанавливали в систему шаблоны каких-либо *конфигураций*, их перечень можно будет найти в данном окне. Нас готовые *конфигурации* в данном курсе не интересуют, поэтому мы выбираем вариант создания *информационной базы* без *конфигурации*. Он предназначен либо для разработки новой *конфигурации*, либо для загрузки в пустую *конфигурацию* выгруженной ранее *информационной базы* или *конфигурации* из файла. Нажав в очередной раз кнопку **Далее,** мы попадаем в следующее окно, которое служит для указания наименования и типа расположения базы

В нашем случае наименованием будет **"Основы разработки"**, тип расположения - **На данном компьютере или на компьютере в локальной сети**. Второй вариант используется в том случае, если вы имеете дело с сетевой версией программы и собираетесь разместить базу на сервере 1С:Предприятия.

Нажав в очередной раз **Далее**, мы попадаем в последнее окно добавления *информационной базы*.

Здесь мы задаем каталог *информационной базы* и язык.

Нажмем **Готово** - будет создана пустая *информационная база*, в списке баз появится название новой базы.

Обратите внимание на то, что по нажатию кнопки **Удалить** выделенная *информационная база* будет удалена лишь из списка стартового окна, но не из системы.

В каталоге только что созданной пустой *информационной базы* есть *файл* 1Cv8.1CD и *папка* 1Cv8Log. *Файл* - это и есть *информационная база*. Сейчас он имеет совсем небольшой размер - 256 Кб. Размер будет расти в ходе разработки *конфигурации* и ввода данных пользователями системы.

Сейчас, после создания новой пустой *конфигурации* мы готовы к первому ее запуску в режиме *конфигуратора*. Выделим ее наименование и нажмем на кнопку **Конфигуратор**. Откроется окно *конфигуратора* - оно будет совершенно пустым. Выполним команду *меню* **Конфигурация > Открыть конфигурацию**. В левой части окна появится *дерево конфигурации*.

Деревом конфигурации вы будете постоянно пользоваться при разработке. Можно заметить, что в окне *дерева конфигурации* уже что-то есть, хотя выше мы создали новую пустую конфигурацию. Дело в том, что здесь представлены лишь пустые группы элементов, которые мы, при работе над нашей учебной конфигурацией, заполним соответствующими *объектами*.

Конфигурация - это описание структуры данных, на основе которых строится работа пользователя с системой в режиме 1С:Предприятие.

Предположим, нам нужно, чтобы *пользователь* мог ввести в систему некий документ, который имеет следующие графы (реквизиты, говоря языком 1С:Предприятия):

- Дата;
- Номер документа;
- Ответственное лицо;
- Сумма выручки;

Для этого мы создаем (описываем) в режиме *конфигуратора объект* **Документ**, указываем набор его реквизитов, задаем типы данных для этих реквизитов, настраиваем экранные формы документа и другие параметры. В итоге, *пользователь* системы сможет работать с документом в режиме 1С:Предприятие.

Нередко, говоря о разработке для *платформы* 1С:Предприятие, применяют термин "*программирование*". "*Программирование*" для 1С:Предприятия - это не только написание программного кода, но и работа в визуальном режиме - создание и настройка *объектов*, разработка экранных форм, работа с различными конструкторами, ускоряющими процесс разработки. Некоторые действия в ходе разработки можно совершить лишь в визуальном режиме.

Прежде чем начинать изучение конфигурирования, опишем практическую ситуацию, которая легла в основу сквозного примера, а так же рассмотрим некоторые основные приемы работы, которые пригодятся нам в дальнейшем.

Мы будем рассматривать основы разработки прикладных *конфигураций* на примере автоматизации деятельности салона красоты "Марина". В салоне работают директор, *администратор*, мастера и другие сотрудники. Основная задача мастеров - оказание услуг клиентам. Салон закупает материалы у поставщиков. Материалы расходуются в ходе оказания услуг, а также их можно продавать клиентам - физическим лицам или организациям.

Мы автоматизируем *деятельность* салона. В частности, нужно автоматизировать учет материалов, учет деятельности мастеров, учет клиентов с возможностью назначения особых условий обслуживания для постоянных клиентов. Нужно автоматизировать расчет заработной платы для сотрудников, создать отчеты, которые позволят руководству анализировать эффективность деятельности салона. В ходе решения задачи мы столкнемся и со многими другими практическими ситуациями, требующими автоматизации.

Мы не будем стремиться к созданию решения, которое можно будет использовать на практике. Наша основная задача - рассмотреть работу с *объектами* системы, использовав избранную предметную область для демонстрации возможностей и особенностей системы. Хотя, с другой стороны, нашу итоговую *конфигурацию*, после доработки, можно будет использовать на практике.

Продолжим знакомство с *Конфигуратором*. А именно, приступим к изучению операций с *объектами*.

Первой операцией с *объектами*, которую мы освоим в *Конфигураторе*, будет создание *объектов*. Создать *объект* нужного вида можно несколькими способами. Например - вызвав контекстное *меню* группы *объектов* и выбрав в нем *пункт* **Добавить**

Обратите внимание на пиктограмму, которая сопровождает *пункт меню*. Такую же пиктограмму имеет кнопка **Добавить** на панели инструментов окна *дерева конфигурации*. Эта же *команда* продублирована в *меню* **Действия**.

Обратите внимание на то, что одно и то же действие в *Конфигураторе* можно выполнить различными способами.

Пока не будем выполнять описанную команду. В следующей лекции мы попрактикуемся в создании *объектов* и в других операциях с ними.

Выше мы использовали понятие " *объект* ", однако, не поясняли его. Если вы знакомы с объектно-ориентированной *методологией программирования*, понятие " *объект* " должно быть вам знакомо. Если нет - давайте рассмотрим простой пример, который позволит понять сущность *объектов*. Представьте себе, что *конфигурация* — это ноутбук. Ноутбук состоит из отдельных частей, связанных воедино, взаимодействующих друг с другом. Это - *монитор*, клавиатура, материнская *плата*, центральный *процессор*, оперативная *память* - *список* можно продолжать очень долго. Всё то, из чего состоит наш ноутбук, его детали - это *объекты*. Каждый *объект* обладает определенной функциональностью. Он может связываться каким-то образом с другими *объектами*, он имеет средства управления, может сообщать другим *объектам* о своем состоянии. В более широком смысле и ноутбук - это тоже *объект*. Им можно управлять, используя клавиатуру. Он может выводить сообщения с помощью монитора. Они составляют *интерфейс*, с помощью которого *пользователь* взаимодействует с ноутбуком.

Итак, *объект*, это "деталь" *конфигурации*. Существуют *объекты* различных видов - их *список* можно видеть в окне *дерева конфигурации*.

Сейчас рассмотрим некоторые другие не менее важные действия, которые выполняют в *Конфигураторе*.

Одна из важнейших сервисных операций, которую вам постоянно придется выполнять - это создание архивной копии *информационной базы*.

Архивные копии рабочих баз нужно делать достаточно регулярно - для того, чтобы обезопасить себя от потери информации. Если вы собираетесь выполнить какую-нибудь операцию, которая может

повлиять на нормальную работу *конфигурации*, например - обновить *конфигурацию* - прежде чем ее выполнять, вы должны обязательно сделать архивную копию *информационной базы*.

Имейте в виду, что операция по архивированию *информационной базы* универсальна, как и многие другие *операции* в *Конфигураторе*. То есть - используя описанные команды вы сможете сделать архивную копию любой *конфигурации*, работающей на *платформе* 1С:Предприятие 8.

Сделать архивную копию *информационной базы* можно несколькими способами.

Первый из них заключается в обычном копировании или архивировании каталога с *информационной базой*.

Второй способ - воспользоваться инструментами архивирования, встроенными в *Конфигуратор*. А именно, если выполнить команду *меню* **Администрирование > Выгрузить информационную базу**, появится окно, которое содержит стандартный *запрос* о задании имени сохраняемого файла. В нашем случае это *файл* с расширением .DT. Он содержит архивную копию *информационной базы*. В случае повреждения или утери *информационной базы* вы можете восстановить ее из архивного файла командой **Администрирование > Загрузить информационную базу**.

Помните, что *информационная база* содержит не только *конфигурацию*, но и базу данных, которая формируется при работе пользователя с системой, то есть - данные, которые он вводит в систему в режиме 1С:Предприятие.

При загрузке *информационной базы* из файла вы получаете то ее состояние, которое она имела на момент выгрузки. Изменения, сделанные после этого, теряются. Предположим, мы выгрузили *информационную базу* и после этого создали новый *объект* системы, ввели какие-то данные в пользовательском режиме. Если после этих действий загрузить ранее выгруженную *информационную базу*, окажется, что сделанные изменения утрачены.

Информационная база хранит две *конфигурации*. Одна из них называется ***основной конфигурацией*** или просто *конфигурацией*. Именно ее мы правим в ходе работы с *конфигуратором* и именно ее открываем командой **Конфигурация > Открыть конфигурацию**. Вторая *конфигурация* называется ***конфигурацией базы данных***. Она используется в ходе работы пользователей. Редактировать *основную конфигурацию* можно в процессе работы пользователей с *информационной базой*. Для того, чтобы перенести изменения в *конфигурацию базы данных*, нужно, чтобы пользователи завершили работу с программой. Для того, чтобы открыть *конфигурацию базы данных*, выполните команду **Конфигурация > Конфигурация базы данных > Открыть конфигурацию БД**. Команды изменения *объектов* в окне *конфигурации базы данных* заблокированы.

Если, открыв *конфигурацию*, мы внесем в нее изменения, например, создадим новый *объект*, в заголовке окна *дерева конфигурации* можно будет наблюдать значок: "*" (признак модифицированности *конфигурации*)

Значок "*" означает, что внесенные изменения не сохранены в *основной конфигурации*. То есть он сигнализирует о том, что изменения, которые мы внесли на этапе конфигурирования, могут быть утеряны при, например, внезапном отключении питания.

Для того, чтобы сохранить *основную конфигурацию*, нужно выполнить команду *меню* **Конфигурация > Сохранить конфигурацию**.

После сохранения значок модифицированности *конфигурации* исчезает, но появляется значок отличия *конфигураций* **"<!>"**

Знак отличия *конфигураций* указывает на то, что изменения, внесенные в *основную конфигурацию*, пока не внесены в *конфигурацию базы данных*.

Для того, чтобы перенести изменения из *основной конфигурации* в *конфигурацию базы данных*, нужно выполнить команду **Конфигурация > Обновить конфигурацию базы данных**. Если изменения, внесенные в *основную конфигурацию* не были сохранены до выполнения этой команды - система, перед обновлением *конфигурации базы данных*, предложит сначала сохранить изменения.

При обновлении *конфигурации базы данных* система выводит окно с перечнем изменений, которые будут внесены в *конфигурацию базы данных* .

Если вы согласны с перечнем изменений, нажмите на кнопку **Принять**, в противном случае - на кнопку **Отмена**.

Признаки отличия *конфигураций* и модифицированности могут отображаться в заголовке окна *дерева конфигурации* одновременно. Если вы внесли изменения в *конфигурацию*, сохранили *основную конфигурацию*, не обновляя *конфигурацию базы данных*, а потом продолжили вносить изменения - оба значка отобразятся в заголовке окна.

Если вы внесли изменения в *основную конфигурацию*, сохранили их, но еще не обновляли *конфигурацию базы данных*, вы можете вернуться к *конфигурации базы данных*, отменив изменения, сделанные в *основной конфигурации*. То есть, фактически, заменить *основную конфигурацию конфигурацией базы данных*. Для этого нужно выполнить команду **Конфигурация > Конфигурация базы данных > Вернуться к конфигурации БД**.

Конфигурацию можно сохранять в *файл* и загружать из файла. Для сохранения *основной конфигурации* в *файл* выполните команду **Конфигурация > Сохранить конфигурацию в файл**. Для

загрузки *конфигурации* из файла выполните команду **Конфигурация > Загрузить конфигурацию из файла**. Выгруженная *конфигурация* хранится в файле с расширением .CF.

В *файл* можно выгружать не только *основную конфигурацию*, но и *конфигурацию базы данных* - это можно сделать командой **Конфигурация > Конфигурация базы данных > Сохранить конфигурацию БД в файл**.

Обратите внимание на то, что выгружая *информационную базу* в *файл*, вы сохраняете в этом файле *информационную базу* целиком - то есть *основную конфигурацию*, *конфигурацию базы данных* и саму базу данных, которая содержит информацию, введенную пользователем в пользовательском режиме.

Сохраняя *конфигурацию* в *файл* вы сохраняете лишь *конфигурацию* - структуру данных, которая используется для описания возможностей, доступных пользователю в режиме 1С:Предприятие.

В процессе правки *конфигурации* в режиме *конфигуратора* бывают моменты, когда нужно проверить функциональность разрабатываемого решения в режиме 1С:Предприятие. *Конфигурацию* можно открыть в этом режиме прямо из *Конфигуратора*, воспользовавшись командой **Сервис > 1С:Предприятие** или соответствующей кнопкой на панели инструментов. При таком способе запуска *программа* будет работать точно так же, как она работала бы, если бы вы запустили ее в режиме 1С:Предприятие из стартового окна программы.

Еще один режим запуска, доступный из *Конфигуратора*, предназначен для отладочных целей. Для того, чтобы открыть *конфигурацию* в режиме отладки, выполните команду **Отладка > Начать отладку** или нажмите соответствующую кнопку.

Если до попытки запуска системы 1С:Предприятие из *Конфигуратора* в *конфигурацию* были внесены изменения, не отраженные в *конфигурации базы данных*, перед запуском система уточнит, хотите ли вы обновить *конфигурацию базы данных*

Если вы ответите **Нет** - *программа* запустится, но изменения из *основной конфигурации* не будут перенесены в *конфигурацию базы данных*. При ответе **Да** система сначала обновит *конфигурацию базы данных*, а потом запустится в режиме 1С:Предприятие.

Выводы

В данной лабораторной работе мы начали знакомство с *платформой* 1С:Предприятие 8. Сейчас вы должны уметь создавать новую *информационную базу*, запускать программу в режиме *конфигуратора* и 1С:Предприятие, открывать *информационную базу*, создавать *объекты*, архивировать *информационную базу* и сохранять *конфигурацию* в *файл*. Так же вы должны уметь запускать 1С:Предприятие из *Конфигуратора* для проверки функциональности разрабатываемого *прикладного решения* в различных режимах - в режиме отладки и в обычном режиме.

2.2 Практическое занятие «Разработка информационной системы для хранения информации о сотрудниках предприятия»

Справочники в 1С:Предприятии 8 используются для организации хранения информации об однотипных объектах. Например, *справочник* **Клиенты** будет хранить данные о клиентах, *справочник* **Сотрудники** - о сотрудниках организации, **Контрагенты** - о контрагентах, **Номенклатура** - о товаро-материальных ценностях, **Подразделения** - о подразделениях организации. *Справочник* можно сравнить с обычной бумажной картотекой. Каждая *конфигурация* обычно содержит множество *справочников*.

Обратите внимание на то, что в режиме конфигурирования мы создаем структуру справочника, определяем формы, которые позволяют работать с ним, устанавливаем его связи с другими объектами конфигурации. А в режиме 1С:Предприятие, то есть - в пользовательском режиме - происходит наполнение справочника информацией. Эта особенность характерна и для других подобных объектов. Их структуру и особенности задает программист, а ввод в систему новых элементов осуществляет пользователь. Стоит отметить, что в случае со справочниками мы все же можем создавать его элементы в режиме конфигурирования. Это - так называемые предопределенные элементы, на которые обычно опираются какие-то алгоритмы разрабатываемой конфигурации. Пользователь не может удалить такие элементы - иначе нарушилась бы логика работы системы.

Давайте создадим новый справочник и рассмотрим особенности работы с ним. Пусть это будет справочник Физические Лица. Как следует из названия, он предназначен для хранения сведений о физических лицах. Мы планируем хранить в этом справочнике персональные сведения о физических лицах.

Для того чтобы создать справочник, перейдем в ветвь Справочники дерева конфигурации и создадим в ней новый элемент.

Окно редактирования объекта конфигурации имеет множество закладок, которые позволяют последовательно настраивать свойства объекта, редактировать его структуру. Обратите внимание на кнопки Назад и Далее, которые расположены в нижней части окна. Они предназначены для прохождения по вкладкам и настройки его свойств в правильной последовательности. Вы можете либо пользоваться этими кнопками, либо выбирать закладки вручную. Рассмотрим вкладки окна редактирования объекта. Сейчас мы не будем стремиться рассмотреть их все - позже, работая над конфигурацией, мы сможем увидеть особенности их использования.

Основные. Здесь расположены уже знакомые вам свойства Имя, Синоним, Комментарий. Они имеют тот же смысл, что и у других объектов.

Данные. Одна из важнейших закладок настройки справочника

Для начала на закладке Данные нужно настроить длину кода и наименования элемента справочника.

Длина кода. 9-значный код способен вместить такое количество элементов, которого гарантированно хватит не только для нашей учебной конфигурации, да и для большинства реально используемых конфигураций. На практике длина кода бывает как меньше, так и больше 9. Код используется для идентификации элементов справочника и код каждого элемента уникален. Система сама следит за уникальностью кодов.

Длина наименования. Это свойство справочника обычно хранит краткое наименование элемента. В нашем случае в справочнике планируется хранить сведения о физических лицах, вполне логично в качестве наименования использовать какую-то комбинацию фамилии, имени и отчества физического лица. Будем использовать сокращенную форму, например, "Иванов И.И.", однако, фамилии могут быть довольно длинными, поэтому увеличим длину наименования до 30 символов.

Обратите внимание на то, что в реквизиты любого справочника входят, как минимум, код и наименование. То есть, если мы на данном этапе работы запустим систему в режиме 1С:Предприятие и попытаемся поработать со справочником, то набор данных для каждого из его элементов будет ограничен кодом и наименованием. На практике могут применяться подобные справочники, например, что-то вроде справочника Должности. Поля Наименование вполне хватает для того, чтобы сохранить название должности, а код используется в служебных целях.

Продолжаем настройку закладки Данные. Здесь можно найти два очень важных поля: Реквизиты и Табличные части.

Реквизиты - в этом поле можно описать дополнительные характеристики элемента справочника. Для того, чтобы добавить в справочник новый реквизит, щелкните по кнопке Добавить в поле Реквизиты или воспользуйтесь соответствующей командой контекстного меню. Добавление нового реквизита сродни добавлению нового объекта - система создает его, присваивает имя по умолчанию и выводит окно его свойств.

Перед нами реквизит Фамилия. Очевидно, что такой реквизит должен хранить фамилию физического лица в виде строки, длину строки можно ограничить 30-ю символами.

Добавим в справочник следующие реквизиты (табл. 3.1):

Таблица 3.1. Реквизиты справочника ФизическиеЛица

Имя реквизита	Тип	Параметры типа
Фамилия	Строка	Длина: 30
Имя	Строка	Длина: 30
Отчество	Строка	Длина: 30
ДатаРождения	Дата	Состав даты: Дата
Пол	ПеречислениеСсылка.Пол	-
Район	СправочникСсылка.Районы	-

В конце таблицы есть пара строк, которые могут вызвать у вас затруднения. Типов данных "Перечисление.Ссылка.Пол" и "Справочник.Ссылка.Районы" в системе нет. Их нужно создать самостоятельно, создав соответствующие объекты. Прежде чем создавать эти объекты, давайте поговорим о данных, которые они будут хранить.

Для указания пола сотрудника мы не случайно выбрали объект Перечисление. Дело в том, что при выборе пола мы ограничены двумя вариантами - либо мужской, либо женский. Почему бы не предоставить пользователю возможность вводить информацию о поле в виде строки, не используя перечисление? Предположим, мы дадим пользователю текстовое поле, в которое предложим ввести пол. Разные пользователи (да и один и тот же пользователь в разное время) введут в это поле разные значения. Это могут быть "М" и "Ж" или "муж." и "жен." или "мужской" и "женский". Можно подобрать и другие варианты. Как видите, в итоге получится, что если мы захотим, например, получить сведения о количестве физических лиц мужского пола, зарегистрированных в нашей базе, нам придется "пробираться" сквозь разнообразные наименования, введенные пользователями при заполнении базы. Такая ситуация весьма нежелательна и неудобна. Поэтому везде, где только можно, следует применять средства, стандартизирующие ввод данных. В случае с закрытым (или редко пополняемым, причем, пополняемым администратором в конфигураторе) списком вариантов следует пользоваться перечислениями.

Реквизит Район мы будем использовать для указания района города, в котором проживает физическое лицо. Этот реквизит администрация автоматизируемого салона красоты будет использовать для

планирования выездной работы мастеров. Для хранения этой информации мы применим справочник. Почему бы нам не использовать перечисление? Дело в том, что набор районов города обычно далек от закрытого. Поэтому логично предоставить пользователям самостоятельно формировать список районов и самостоятельно контролировать правильность его заполнения. Почему бы, в таком случае, не сделать реквизит Район в виде обычной текстовой строки? Дело, снова, в том, что пользователи могут назвать один и тот же район по-разному, в итоге, возникнут сложности с автоматической обработкой данных о районах. То есть, например, если нам нужно будет выполнить задачу "подобрать мастеров, которые могут обслужить клиента, проживающего в районе Х", при использовании справочника Районы мы указываем системе конкретный элемент справочника, соответствующего району проживания клиента, а система подбирает по этому району мастеров. Если же сделать ввод данных о районах произвольным, неразбериха в базе гарантирована.

Создадим два новых объекта - перечисление Пол и справочник Районы.

Добавим два значения для перечисления: "Мужской" и "Женский". Окно свойств значения перечисления поддерживает лишь ввод имени, синонима и комментария, других свойств у значения перечисления нет. Перечисление может иметь формы, которые используются для вывода информации, макеты, которые применяются для формирования печатных форм.

Кстати, обратите внимание на окно Свойство - его можно использовать вместе с окном редактирования объекта конфигурации для правки свойств справочника.

Сейчас пришло время опробовать только что созданный справочник ФизическиеЛица на практике. Запускаем 1С:Предприятие с нашей конфигурацией.

Выполним команду меню Операции > Справочник. Появится окно выбора справочника

Обратите внимание на то, что в окне Выбор объекта: Справочник мы видим синоним справочника, а не его имя. В реально используемых конфигурациях имя и синонимы справочников нередко различаются весьма значительно. Подобные окна появляются и при выборе многих других пунктов меню Операции. Выберем нужный справочник - Физические лица . Сделаем по строке с его наименованием двойной щелчок или, выделив, нажмем ОК, или, опять же, выделив, нажмем клавишу Enter на клавиатуре.

В настоящий момент справочник пуст. Мы можем видеть лишь строку, которая содержит заголовки, соответствующие реквизитам справочника. Создадим новый элемент справочника, для этого воспользуемся кнопкой Добавить, выберем пункт меню Действия > Добавить или просто нажмем кнопку Ins на клавиатуре. Одного и того же результата можно добиться различными путями, это характерно для многих операций - учитывайте это при работе в режиме 1С:Предприятие.

Код присваивается элементу справочника автоматически. В данном случае мы можем его редактировать, но делать этого не будем. Эту возможность, кстати, весьма разумно заблокировать (или разработать особую процедуру смены номера, которая исключает случайную смену номера и т.д.).

В полях Наименование, Фамилия, Имя, Отчество, содержится, в соответствии со свойствами реквизитов справочника, обычные строковые значения. Мы можем вводить их вручную с клавиатуры. Если, предположим, мы вводим сведения о некоем Иванове Иване Ивановиче, то понятно, что содержимое поля Наименование (а там, при таком предположении должно быть Иванов И.И.) вполне можно сформировать автоматически на основе данных, введенных в поля Фамилия, Имя, Отчество. Эту удобную возможность мы реализуем позже, это - одна из многочисленных доработок справочника Физические лица.

Когда мы заполняем поле Дата рождения, мы можем либо ввести дату с клавиатуры (в формате ДД.ММ.ГГГГ), либо, что обычно удобнее, ввести ее с помощью кнопки Выбрать, расположенной в правой части окна.

Заполняя поле Пол, мы нажимаем кнопку с тремя точками (она тоже называется Выбрать) и выбираем нужное значение из выпадающего списка. Заполняя поле Район, мы, нажав на кнопку Выбрать, видим окно справочника Районы. Прежде чем мы сможем указать район в справочнике Физические лица, мы должны внести его в справочник Районы.

Создаем новый элемент справочника Районы, пусть это будет район с наименованием "Ленинский".

Когда ввод информации в элемент справочника Районы завершен, нажмем кнопку ОК в форме элемента справочника. Этот элемент появится в окне Справочник Районы, после чего мы сможем выбрать его двойным щелчком для подстановки в поле Район редактируемого элемента справочника Физические лица. В табл. 3.2 вы можете видеть состав данных созданного элемента справочника Физические лица.

Таблица 3.2. Данные, введенные в элемент справочника Физические лица

Реквизит	Содержимое
Наименование	Иванов И.И.
Фамилия	Иванов
Имя	Иван
Отчество	Иванович
Дата рождения	27.02.1984
Пол	Мужской

Район Ленинский

Заполнив поля окна редактирования элемента справочника, нажмем кнопку ОК. Данные будут записаны в базу, элемент появится в окне Справочник Физические лица.

Что еще мы можем сделать с элементом справочника на данном этапе работы? Осмотримся в окне списка справочника, в частности, посмотрим на меню Действия, выделив элемент справочника.

Действие большинства команд, которые перечислены в меню Действия вполне понятно из их названий. Однако некоторые из них нуждаются в пояснениях.

Команда Удалить непосредственно позволяет удалить текущий элемент справочника непосредственно. При непосредственном удалении элемента справочника возможно нарушение ссылочной целостности информационной базы.

Команда Установить пометку удаления лишь помечает объект на удаление, не производя его удаления из базы.

Создадим в справочнике Районы еще одну запись - добавим туда район, который называется "Ненужный район". Пометим на удаление все элементы справочника. В данный момент это - "Ненужный район" и "Ленинский".

Для того, чтобы попытаться удалить помеченные объекты, следует выбрать пункт меню Операции > Удаление помеченных объектов.

В появившемся окне следует запустить проверку нажатием на кнопку Контроль.

После проведения контроля станет известно, какие объекты можно удалить, а какие – нельзя.

Система сообщает нам о том, какие из помеченных объектов можно удалить (они отмечены зеленой галочкой) а какие - нет (они отмечены красной галочкой). Здесь же, выделив мышью удаляемый объект, можно узнать, где на него есть ссылки. Нажав на кнопку Удалить мы удаляем те объекты, которые ни к чему не "привязаны".

Нельзя удалить те объекты, которые привязаны к другим объектам. В нашем случае элемент справочника Физические лица содержит ссылку на элемент справочника Районы. Физическому лицу Иванову И.И. сопоставлен район Ленинский. Теперь если мы захотим удалить район Ленинский, система не даст нам этого сделать до тех пор, пока этот район "привязан" к одному из физических лиц.

Если вызвать меню Действия для объекта, помеченного на удаление, в нем можно будет найти пункт Снять пометку удаления.

Кнопка Вывести список позволяет выводить справочник в виде, подходящем для распечатки или сохранения. Сразу после выполнения этой команды появится окно, которое позволяет, во-первых, указать формат выходного документа (табличный документ или текстовый документ), а во-вторых - выбрать состав данных справочника, которые в этот документ выгружаются.

Кстати, вы обратили внимание на то, что наши справочники оснащены формами, а мы, работая в конфигураторе, никаких форм не создавали? Если да - тогда вам будет интересно узнать о том, как управлять поведением форм объектов. К тому же, напомним, что мы, работая в справочнике Физические лица, решили, что в нем не помешает настроить автоматическое заполнение поля Наименования на основе полей Фамилия, Имя, Отчество.

2.3 Практическое занятие «Разработка основных документов, отчеты»

В предыдущей лабораторной работе мы создали документ ПоступлениеМатериалов, который используется для отражения в учетной системе информации о поступивших материалах. Работа над документом еще не завершена. В частности, обычно документы имеют не только электронные, но и печатные представления.

В реальных учетных системах, построенных на базе 1С:Предприятие 8 работа с некоторыми документами строится следующим образом. Документ заполняют, сохраняют, распечатывают. После распечатки выполняются какие-либо действия, предусмотренные документом, на печатной форме документа собираются необходимые подписи, после чего документ, уже электронный, открывают, проверяют соответствие электронной и печатной версий и проводят.

Для создания печатной формы документа используют макеты. Макет - это объект конфигурации, который чаще всего используется для хранения табличных документов, на основе которых создаются печатные формы. Процесс разработки печатной формы состоит из двух этапов.

На первом этапе происходит разработка макета. Макеты обычно представляют собой табличные документы, однако, можно работать с макетами других видов. В табличном документе задаются именованные области, настраиваются параметры этих областей, после чего данные области применяют для "сборки" печатной формы документа.

Второй этап разработки печатной формы - это создание программного кода, который на основе данных документа и макета "собирает" печатную форму.

Разработку печатных форм можно ускорить, воспользовавшись так называемым конструктором печати. Для этого откроем в конфигураторе окно редактирования документа ПоступлениеМатериалов, в

этом окне выберем вкладку Макеты, на ней нажмем кнопку Конструкторы, в выпавшем меню выберем пункт Конструктор печати.

Появившееся окно конструктора печати, проведет вас через несколько шагов, необходимых для создания печатной формы. А именно, на первом шаге нужно указать имя процедуры печати и ее расположение. По умолчанию процедура называется Печать, расположена она будет в модуле объекта. Укажем модуль формы документа для хранения процедуры и нажмем на кнопку Далее.

Во втором окне конструктора нам предлагается задать, какие реквизиты документа попадут в шапку печатной формы, то есть - в ее верхнюю часть.

В правой части формы, которая называется Реквизиты документа можно найти список реквизитов документа, в левой части, которая представляет собой список реквизитов шапки, отображаются реквизиты документа, перенесенные туда либо двойным щелчком, либо - с помощью кнопки с изображением стрелки. Для того, чтобы перенести все реквизиты в шапку - достаточно нажать на кнопку с двумя стрелками. Перенесем в шапку печатной формы все реквизиты документа кроме реквизита СтоимостьМатериалов. Его мы прибережем для того, чтобы вывести в подвале - то есть - в нижней части печатной формы.

Нажав в очередной раз кнопку Далее, мы попадаем в окно конструктора печати, которое предназначено для настройки вывода в печатную форму табличной части документа. Логика работы с данным окном ничем не отличается от работы с вышеописанным. На рис. 6.4 вы можете видеть состояние третьего окна конструктора печати после того, как мы настроили его таким образом, чтобы на печать выводились все доступные реквизиты табличной части документа.

Следующее, четвертое, окно конструктора, содержит набор полей для вывода в подвале. Выберем для отображения в подвале реквизит СтоимостьМатериалов.

Очередное нажатие кнопки Далее приводит нас к последнему окну конструктора. Здесь мы укажем, что хотим, чтобы в форму документа была добавлена кнопка Печать.

Кнопка ОК завершает работу конструктора. Сразу после этого на экране будет отображено окно редактирования макета и окно редактора форм.

Обратите внимание на то, что на нижней панели управления формы появилась новая кнопка - Печать. Обратите так же внимание на то, что табличный документ, который представляет собой макет, разбит на именованные области. Здесь мы можем видеть области Заголовок, Шапка, МатериалыШапка, Материалы и Подвал. В этих областях мы можем видеть ячейки, содержащие обычный текст, без каких-либо специальных символов. При выводе именованной области в печатный документ, эти тексты будут выведены в том виде, в котором они представлены в макете, с использованием того же форматирования. Если же в макет выводится ячейка, текст в которой обрамлен значками "<>", в этой ячейке будет выведен какой-либо реквизит документа. В частности, в угловых скобках в данном случае выводятся параметры для вывода их в ячейках. Имена параметров совпадают с соответствующими именами реквизитов документа.

То, что вы сейчас узнали, сделает достаточно простым знакомство с процедурой печати, которая была сформирована автоматически. Напомним, что при работе конструктора печати мы указали, что хранить эту процедуру следует в модуле формы. Найдем процедуру в модуле формы (вкладка Модуль). Вот как выглядит текст модуля:

```
Процедура Печать(Элемент)
    //{{_КОНСТРУКТОР_ПЕЧАТИ_ЭЛЕМЕНТ(Печать)
    // Данный фрагмент построен конструктором.
    // При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!
    ТабДок = Новый ТабличныйДокумент;
    Макет = Документы.ПоступлениеМатериалов.ПолучитьМакет("Печать");
    // Заголовок
    Область = Макет.ПолучитьОбласть("Заголовок");
    ТабДок.Вывести(Область);
    // Шапка
    Шапка = Макет.ПолучитьОбласть("Шапка");
    Шапка.Параметры.Заполнить(ЭтотОбъект);
    ТабДок.Вывести(Шапка);
    // Материалы
    Область = Макет.ПолучитьОбласть("МатериалыШапка");
    ТабДок.Вывести(Область);
    ОбластьМатериалы = Макет.ПолучитьОбласть("Материалы");
    Для Каждого ТекСтрокаМатериалы Из Материалы Цикл
        ОбластьМатериалы.Параметры.Заполнить(ТекСтрокаМатериалы);
        ТабДок.Вывести(ОбластьМатериалы);
    КонецЦикла;
    // Подвал
    Подвал = Макет.ПолучитьОбласть("Подвал");
    Подвал.Параметры.Заполнить(ЭтотОбъект);
    ТабДок.Вывести(Подвал);
```

```
    ТабДок.ОтображатьСетку = Ложь;
    ТабДок.Защита = Ложь;
    ТабДок.ТолькоПросмотр = Ложь;
    ТабДок.ОтображатьЗаголовки = Ложь;
    ТабДок.Показать();
    //}}_КОНСТРУКТОР_ПЕЧАТИ_ЭЛЕМЕНТ
КонецПроцедуры
```

Поясним программные конструкции, которые были использованы в данной процедуре

ТабДок = Новый ТабличныйДокумент;

В переменную ТабДок записываем ссылку на новый объект типа ТабличныйДокумент. Это именно тот документ, который, после работы процедуры, будет содержать готовую печатную форму. Сразу после создания табличный документ не виден пользователю, то есть все действия, которые мы с ним будем производить дальше, выполняются с невидимым документом. После завершения программного формирования документа, мы сделаем его видимым.

Макет = Документы.ПоступлениеМатериалов.ПолучитьМакет("Печать");

В переменную Макет записываем ссылку на макет с именем Печать, который принадлежит нашему документу. Будем использовать эту переменную для доступа к макету.

```
// Заголовок
    Область = Макет.ПолучитьОбласть("Заголовок");
    ТабДок.Вывести(Область);
```

В этом участке кода мы сначала получаем именованную область макета, которая носит имя Заголовок. Фактически, метод ПолучитьОбласть объекта Макет возвращает новый табличный документ, который представляет собой область, полученную по имени или другим способом. Ссылка на документ записывается в переменную Область. Используя метод Вывести объекта ТабДок мы выводим полученную область в наш табличный документ.

```
// Шапка
    Шапка = Макет.ПолучитьОбласть("Шапка");
    Шапка.Параметры.Заполнить(ЭтотОбъект);
    ТабДок.Вывести(Шапка);
```

В заголовке, если посмотреть на макет, есть лишь текст, содержащий название документа и никаких полей, которые нужно заполнять данными документа. А вот в шапке такие параметры имеются. Область, соответствующую шапке, мы получаем уже рассмотренным методом. А вот следующая команда - Шапка.Параметры.Заполнить(ЭтотОбъект); позволяет заполнить ячейки области, содержащие параметры, реквизитами объекта, переданного в качестве параметра (в данном случае мы имеем дело с объектом ЭтотОбъект - он позволяет получить тот документ, с которым мы в данный момент работаем). Соответствия между параметрами и реквизитами документа устанавливаются по именам. Ячейка, содержащая параметр <Номер> будет заполнена реквизитом документа с именем Номер и так далее.

```
// Материалы
    Область = Макет.ПолучитьОбласть("МатериалыШапка");
    ТабДок.Вывести(Область);
```

В данном участке кода мы получаем область, соответствующую шапке таблицы и выводим ее в табличный документ. Далее следует обработка таблицы и ее построчный вывод:

```
ОбластьМатериалы = Макет.ПолучитьОбласть("Материалы");
    Для Каждого ТекСтрокаМатериалы Из Материалы Цикл
            ОбластьМатериалы.Параметры.Заполнить(ТекСтрокаМатериалы);
            ТабДок.Вывести(ОбластьМатериалы);
    КонецЦикла;
```

Здесь мы видим цикл, который перебирает каждую строку табличной части Материалы. На основе каждой из строк (переменная ТекСтрокаМатериалы) он заполняет параметры области Материалы и выводит ее в табличный документ.

```
// Подвал
    Подвал = Макет.ПолучитьОбласть("Подвал");
    Подвал.Параметры.Заполнить(ЭтотОбъект);
    ТабДок.Вывести(Подвал);
```

Подвал выводится уже знакомым вам образом. Далее настраиваются некоторые свойства формируемого документа, влияющие на его внешний вид:

```
ТабДок.ОтображатьСетку = Ложь;
    ТабДок.Защита = Ложь;
    ТабДок.ТолькоПросмотр = Ложь;
    ТабДок.ОтображатьЗаголовки = Ложь;
    ТабДок.Показать();
```

Здесь отключается отображение сетки, обычной для табличных документов, устанавливаются в Ложь свойства документа, отвечающие за его защиту и за запрет редактирования, отключается отображение заголовков, и, после того как все настроено, табличный документ делается видимым - с помощью метода Показать.

Запустим наше прикладное решение в режиме 1С:Предприятие и посмотрим, как выглядит печатная форма для какого-нибудь документа, ПоступлениеМатериалов.

Для того чтобы распечатать эту форму, достаточно воспользоваться командой главного меню программы Файл > Печать.

Выше, говоря об устройстве макета, мы рассмотрели лишь результаты работы конструктора печати. Вы можете видеть то, что получилось на выходе конструктора печати, позволяет получить вполне стандартно выглядящую печатную форму. Если нам нужно что-то большее - как в плане внешнего вида формы, так и в плане использования параметров для ее заполнения, нам нужно будет воспользоваться ручным редактированием (или созданием с нуля) макетов и модулей печати.

На данном этапе не будем больше улучшать и дорабатывать документ ПоступлениеМатериалов. Займемся еще одним документом нашей конфигурации.

В соответствии с логикой нашей учетной системы, мы создали документ, который отражает поступление материалов, которые хранятся у определенного материально ответственного лица - сотрудника организации. Теперь создадим документ, который используется для отпуска материалов, вверенных материально-ответственным лицам. Назовем его ОтпускМатериаловМастеру.

Логика бизнес-процесса организации, который мы собираемся автоматизировать с помощью данного документа, такова. К сотруднику, за которым закреплены материалы (назовем его "кладовщик") обращается другой сотрудник (назовем его "мастер"), которому материалы нужны для проведения каких-либо работ. В заказе мастера есть количество требуемых материалов. На основании требования мастера, кладовщик должен передать ему требуемое количество материалов - но не больше, чем за ним числится - и выдать документ, который содержит информацию о количестве и стоимости выданных материалов. Если у одного из кладовщиков не хватает материалов для выдачи мастеру - он может выполнить заказ в пределах своих возможностей, а мастер, в свою очередь, может воспользоваться услугами нескольких кладовщиков.

Материалы отпускают по средней стоимости. Наш документ должен иметь печатную форму, он так же должен проводиться по регистру ОстаткиМатериалов, отражая в нем факт выбытия материалов.

Документ будет иметь следующие реквизиты, табл. 6.1:

Таблица 6.1. Реквизиты документа ОтпускМатериаловМастеру

Имя реквизита	Тип	Параметры типа
ПолучательМатериалов	СправочникСсылка.Сотрудники	
ОтветственныйСотрудник	СправочникСсылка.Сотрудники	
СтоимостьМатериалов	Число	Длина 15, точность 2

Документ будет иметь табличную часть Материалы, которая будет иметь следующие реквизиты:

Таблица 6.2. Реквизиты табличной части Материалы документа ОтпускМатериаловМастеру

Имя реквизита	Тип	Параметры типа
Номенклатура	СправочникСсылка.Номенклатура	
Количество	Число	Длина 10, точность 3
Сумма	Число	Длина 10, точность 2

Напомним, что если какие-либо объекты конфигурации имеют реквизиты, схожие с теми, которые нужны в новых объектах - их можно копировать - это ускоряет работу.

Прежде чем заниматься дальнейшей работой над документом, нам нужно модифицировать регистр накопления ОстаткиМатериалов. А именно, мы введем в него новый реквизит ОтпущеноМастеру с типом СправочникСсылка.Сотрудники. В этом реквизите мы будем хранить справочные сведения о том, какому именно сотруднику были отпущены материалы. Запись в этот реквизит будет осуществляться документом ОтпускМатериаловМастеру.

Не забудем добавить документ ОтпускМатериаловМастеру в список регистраторов регистра ОстаткиМатериалов, установив галочку напротив наименования документа на вкладке Регистраторы окна редактирования свойств регистра.

При вводе информации в табличную часть документа, мы будем заполнять лишь реквизиты Номенклатура и Количество. Сумма для списания будет вычисляться по нажатию специальной кнопки. В частности, при нажатии на эту кнопку будет проверяться, достаточно ли материалов каждого вида числится

за ответственным лицом, и, если достаточно - будет рассчитываться средняя стоимость материалов, которые предполагается списать.

Добавим на командную панель формы новую кнопку. Для этого щелкнем мышью по схематичному изображению кнопки в правой части командной панели, в появившемся окне редактирования свойств кнопки введем ее имя - Рассчитать, в поле Действие нажмем на кнопку Открыть. В модуле формы будет создана процедура, которая будет выполняться при нажатии на кнопку.

Теперь заблокируем поле табличной части Сумма от изменений, установим его свойство ТолькоПросмотр. Поле мы собираемся рассчитывать по нажатию на кнопку Рассчитать и заполнять программно.

Если поразмышлять над тем, как реализовать расчет средней стоимости, то, основываясь на тех знаниях, которые у вас уже есть, можно прийти к выводу о том, что нужно выполнить следующую последовательность действий:

Поочередно перебрать строки табличной части;

Получить для номенклатурной позиции, упомянутой в строке, сведения о количестве и сумме из регистра ОстаткиМатериалов ;

Вычислить стоимость единицы материала (делением общего суммового остатка на общий количественный остаток);

Вычислить стоимость списываемых материалов;

Записать в поле Сумма табличной части результат вычисления.

Пожалуй, из всех вышеперечисленных операций наибольшую сложность для вас может составить работа с регистром накопления. Напомним, что регистр ОстаткиМатериалов - это регистр накопления, вид регистра - Остатки. Для такого регистра можно получить значение остатков (ресурсов) по выбранным измерениям. Вот как выглядит процедура расчета средней стоимости материалов, введенных в табличную часть документа пользователем.

```
Процедура ДействияФормыРассчитать(Кнопка)
        ОстМатериалов=РегистрыНакопления.ОстаткиМатериалов;
        Фильтр=Новый Структура;
        Фильтр.Вставить("ЦентрОтветственности", ОтветственныйСотрудник);
        Для Каждого ТекСтрокаМатериалы Из Материалы Цикл
                Фильтр.Вставить("Номенклатура", ТекСтрокаМатериалы.Номенклатура);
                ТабОстатков = ОстМатериалов.Остатки(,Фильтр,,);
                ТекСтрокаТабОстатков = ТабОстатков.Получить(0);
    ТекСтрокаМатериалы.Сумма=(ТекСтрокаТабОстатков.Сумма/ТекСтрокаТабОстатков.Количество)
*ТекСтрокаМатериалы.Количество;
        КонецЦикла;
СтоимостьМатериалов = Материалы.Итог("Сумма");
КонецПроцедуры
```

Прежде чем рассматривать процедуру, следует дать некоторые пояснения. Здесь мы пользуемся объектом типа РегистрНакопленияМенеджер, доступ к которому обеспечивается с помощью свойства глобального контекста РегистрыНакопления. То есть, другими словами, для того, чтобы получить доступ к регистру накопления ОстаткиМатериалов, определенному в конфигурации, нам следует обратиться к нему с помощью конструкции РегистрыНакопления.ОстаткиМатериалов. Эта конструкция имеет тип РегистрНакопленияМенеджер.

У менеджера регистра накопления (ОстМатериалов) есть различные методы, один из которых - а именно - Остатки() - предназначен для получения остатков. Причем, полный вызов этого метода выглядит так:

Остатки(<Момент времени>, <Отбор>, <Измерения>, <Ресурсы>)

Нас, в данном случае, интересует получение текущих остатков, то есть, остатков на наибольшую дату движения регистра. Поэтому параметр Момент времени мы не используем. Параметры Измерения и Ресурсы мы так же не используем - нам нужно получить остатки по всем измерениям и ресурсам в рамках, заданных параметром Отбор. Параметр Отбор для нас важен, так как мы собираемся получать данные об одном из видов номенклатуры, числящемся за материально ответственным лицом. Параметр Отбор имеет тип Структура. Структура - это набор ключей и соответствующих им значений. В нашем случае структура должна содержать имена измерений регистра, по которым мы хотим получить остатки, и значения, которые нужно отобрать среди указанных измерений. Имена задаются в виде строк, значения имеют тот же тип, что и тип соответствующих измерений.

Данные, полученные из регистра накопления, собраны в таблице значений. Это - таблица, которая имеет несколько строк (в нашем случае это одна строка, нумерация строк начинается с 0) и столбцов. Имена столбцов соответствуют данным, которые хранятся в таблице. В нашем случае таблица будет, кроме прочих, иметь столбцы Сумма и Количество, которые и хранят нужные нам данные.

Теперь рассмотрим процедуру.

ОстМатериалов=РегистрыНакопления.ОстаткиМатериалов;

Записываем в переменную ОстМатериалов ссылку на менеджер регистра ОстаткиМатериалов.

Фильтр=Новый Структура;

В переменную Фильтр запишем ссылку на только что созданную пустую структуру.

Фильтр.Вставить("ЦентрОтветственности", ОтветственныйСотрудник);

Мы планируем получать остатки различных материалов по одному ответственному сотруднику, который указан в шапке документа. Поэтому сразу же вставляем в структуру пару - ключ - "ЦентрОтветственности", значение - ОтветственныйСотрудник.

Обратите внимание на то, что остальные операции процедуры выполняются в цикле.

Для Каждого ТекСтрокаМатериалы Из Материалы Цикл

Открываем цикл перебора строк табличной части Материалы, записывая ссылку на текущую строку в переменную ТекСтрокаМатериалы.

Фильтр.Вставить("Номенклатура", ТекСтрокаМатериалы.Номенклатура);

Вставляем в структуру еще одну пару Ключ - Значение. Она будет применяться для отбора по номенклатуре, указанной в текущей строке табличной части. Метод Вставить у структур действует следующим образом. Если ключа, указанного при вызове метода, еще нет в структуре, он создается. Если есть - его значение заменяется на то, которое передано при вызове метода. Мы вызываем этот метод столько раз, сколько строк есть в табличной части документа. Получается, что лишь первый вызов создает в структуре ключ "Номенклатура". Последующие вызовы лишь меняют его значение на номенклатуру, указанную в строке.

ТабОстатков = ОстМатериалов.Остатки(,Фильтр,,);

В переменную ТабОстатков попадает результат работы метода Остатки(), который выполняется для регистра накопления ОстаткиМатериалов. В результате в переменной ТабОстатков окажется таблица значений, состоящая из одной строки, а эта строка будет содержать значение остатков материалов указанного ранее вида по ответственному сотруднику.

ТекСтрокаТабОстатков = ТабОстатков.Получить(0);

В переменную ТекСтрокаТабОстатков будет записан результат выполнения метода таблицы значений Получить(). В качестве параметра ему передается номер строки.

ТекСтрокаМатериалы.Сумма=(ТекСтрокаТабОстатков.Сумма/ТекСтрокаТабОстатков.Количество)

*ТекСтрокаМатериалы.Количество;

В этом выражении мы сначала получаем среднюю стоимость единицы материалов - делим сумму стоимостного остатков материалов на количество единиц материала. После этого умножаем найденную среднюю стоимость на количество материалов, указанное в строке табличной части и записываем полученное значение в колонку Сумма текущей строки.

КонецЦикла;

Закрываем цикл.

СтоимостьМатериалов = Материалы.Итог("Сумма")

Выводим в реквизит шапки документа итоговую стоимость списываемых материалов.

Проверим работу механизма на практике.

На этом мы завершим разработку формы и ее механизмов, хотя, безусловно, здесь можно сделать еще множество доработок, повышающих удобство работы пользователя, предотвращающих возникновение ошибок. Вы можете проделать эти работы самостоятельно.

Напомним, что документ ОтпускМатериаловМастеру пока не формирует движений по регистру ОстаткиМатериалов. Создадим процедуру проведения документа, воспользовавшись конструктором движений регистров. Обратите внимание на то, что документ фиксирует расход материалов (тип движения регистра - Расход).

Процедура, сформированная конструктором, списывает материалы, не обращая внимание на их фактическое наличие. Реализация механизма контроля возможности проведения документа в зависимости от фактического наличия материалов, которые нужно списать этим документом, будет рассмотрена в следующих лекциях.

И, наконец, создадим печатную форму документа, воспользовавшись конструктором печати. Проверим правильность работы механизмов.

После проведения документа в регистре накопления появились движения с видом движения Расход.

В нашей конфигурации теперь два документа, а это значит, что пришло время научиться пользоваться объектом конфигурации Журнал документов. В журналах документов обычно собирают документы, относящиеся к какому-то одному направлению деятельности организации. В итоге, открыв один журнал документов можно быстро просмотреть все документы по этому направлению. В нашем случае это будет направление автоматизации работы кладовщиков. Создадим новый журнал документов в ветви Журналы документов дерева конфигурации. Зададим имя журнала СкладскиеДокументы.

Перейдем на вкладку Данные окна редактирования свойств журнала. В поле Регистрируемые документы добавим документы ПоступлениеМатериалов и ОтпускМатериаловМастеру.

Теперь, если мы откроем конфигурацию в режиме 1С:Предприятие и выполним команду меню Операции > Документы, в нем появится новый журнал документов Складские документы. Журнал объединил два вида документов.

Как видите, состав граф журнала предоставляет нам информацию лишь о виде документа, о дате и о номере. Если есть необходимость в том, чтобы журнал отображал дополнительные графы, нужно добавить эти графы в конфигураторе и настроить их состав. Обратите внимание на то, что включать и отключать отображение уже существующих граф в журнале можно, воспользовавшись командой Настройка списка меню Действия журнала.

Если попытаться создать новый элемент журнала, нажав на кнопку Добавить в его командной панели, появится окно Выбор вида документа, в котором будут перечислены виды документов, входящих в журнал.

Для создания документа, нужно выбрать его вид и нажать кнопку ОК в окне Выбор вида документа.

Улучшим журнал. Добавим в него две новые графы. Для этого перейдем в режим конфигурирования, откроем окно редактирования свойств объекта СкладскиеДокументы на вкладке Данные.

В поле для добавления граф добавим новую графу, в открывшемся окне ее свойств введем имя - ОтветственныйСотрудник, в поле Ссылки нажмем кнопку с тремя точками - появится окно Выбор объекта: Реквизит. В этом окне отметим реквизиты документов, которые должны отображаться в графе.

В нашем случае выбраны реквизиты ОтветственныйСотрудник. Каждый документ может отобразить в графе журнала лишь один из своих реквизитов.

Точно так же создадим новую графу с именем СтоимостьМатериалов, настроим отображение в ней реквизитов документов СтоимостьМатериалов.

Выводы.

Конфигурация, над которой мы работаем, растет и развивается. Мы создали очередной документ, разработали журнал документов, познакомились с макетами. В нашей информационной базе начинают накапливаться данные.

2.4 Практическое занятие «Разработка запросов»

Одна из функций учетной системы - предоставление пользователям различной информации. Как правило, делается это с помощью отчетов. Например, в нашем случае вполне логично было бы иметь отчет, который выводит информацию о поступивших и выбывших материалах *по* отдельным материально-ответственным лицам (а может быть и *по* всей организации в целом), а так же - об остатках материалов. Подобная функциональность - то есть - *выборка* данных, осуществляется в 1С:Предприятии с помощью *запросов*.

Запросы создают с некоторой целью. Например, она может звучать так: "Узнать количество и *стоимость* материалов, числящихся за Ивановым И.И.". После того, как цель *запроса* сформулирована, нужно выполнить определенные шаги, которые позволяют получить нужную информацию:

1. Подобрать подходящие источники данных для *запроса* ;
2. Составить текст *запроса* - либо вручную, либо пользуясь конструктором *запросов* ;
3. Выполнить *запрос*;
4. Обработать результаты *запроса*.

Прежде чем переходить к практической работе с *запросами*, обсудим общие положения, важные для дальнейшего понимания материала.

Источники данных для запросов

При работе с *запросами* возникает такое понятие, как источник данных для *запроса*. То есть - те места, откуда *запрос* будет брать данные. Источники данных делятся на две группы. Первая - это так называемые *реальные таблицы*. Вторая - *виртуальные*.

Реальные таблицы называются так потому, что они физически хранятся в базе данных. *Реальные таблицы*, в свою очередь, подразделяются на *объектные* (ссылочные) и *необъектные* (не ссылочные).

В *объектных таблицах* хранятся данные объектов системы, то есть - *ссылочных типов данных*. Это - документы, справочники. Эти таблицы имеют поле **Ссылка**, которое содержит ссылку на объект, данные которого представлены в таблице.

В *необъектных таблицах* хранятся данные других типов - например - записи регистров.

Виртуальные таблицы, в отличие от *реальных*, нигде специально не хранятся. Система "собирает" эти таблицы из реальных данных, используя одну или несколько *реальных таблиц*. При создании *виртуальных таблиц* их можно параметризовать, то есть задать параметры, которые ограничивают отбор данных в эти таблицы. Если вы пользуетесь *виртуальными таблицами* (а без них вы вряд ли сможете обойтись, как вы увидите позже), и вам нужно, чтобы они включали в себя данные, ограниченные некоторым отбором, нужно выполнять этот отбор, используя параметры *виртуальных таблиц*. Есть и другие способы выбора из *виртуальных таблиц* нужных данных, но они уступают в скорости работы параметризации этих таблиц.

Поля таблицы могут содержать либо какие-то данные, либо - *вложенные таблицы*. Причем, поле таблицы может иметь какой-то один тип, либо - составной тип данных. Однако, если поле хранит данные, они всегда какого-то одного типа.

Перед созданием *запроса*, или в процессе создания, нужно определиться с источниками данных для него. После того, как источники данных определены, следует написать текст *запроса*.

Написание текста запроса

Во встроенном языке системы есть объект **Запрос**. Именно он используется для работы с *запросами*. Особенности получения данных определяет текст *запроса*. Этот текст можно либо написать вручную, используя конструкции языка, либо воспользоваться так называемым конструктором *запросов*. Конструктор *запросов* позволяет в наглядном виде настроить *запрос*, однако, его результатом является точно такой же текст, который пишут вручную. В *запрос* можно передавать параметры, делать это нужно до выполнения *запроса*.

Выполнение запроса и обработка результатов запроса

После того, как создан текст *запроса* и в *запрос* переданы параметры, *запрос* выполняют. Результат выполнения *запроса* выглядит в виде таблицы, содержащей запрошенные данные. Эту таблицу необходимо обработать и применить полученные данные по назначению - вывести их в отчет, использовать для проверки каких-либо ограничений и так далее.

Для того, чтобы начать практическую работу с *запросами*, мы немного отвлечемся от, собственно, *запросов*, и создадим небольшую внешнюю *обработку*, которая позволит нам в максимально наглядном виде освоить основы работы с *запросами*.

Внешние отчеты и *обработки* удобно использовать для того, чтобы выполнять предусматриваемые ими действия в различных системах. Так же внешние отчеты и *обработки* обычно используют для того, чтобы добавить в существующую конфигурацию какие-то новые возможности, настроить ее под нужды клиента, но при этом не изменять *основную конфигурацию*.

Войдем в **Конфигуратор** и выполним команду **Файл > Новый > Внешняя обработка**.

Введем в *поле* имя **КонсольЗапросов**, для начала редактирования формы *обработки* нажмем на кнопку **Открыть** в *поле* **Основная форма внешней обработки**. Появится окно **Конструктор формы обработки**, оставим в нем все *по* умолчанию и нажмем **Готово**.

Добавим в форму элемент управления **Поле текстового документа**, зададим ему имя **ТекстЗапроса**, установим в параметре **Расширение** *значение* **Язык запросов**. Добавим в *список* реквизитов формы новый *реквизит*, зададим ему имя **ТекстЗапросов** и тип **Строка**.

Это позволит использовать в данном *поле*, при работе в режиме 1С:Предприятие, *конструктор запросов*, синтаксические конструкции языка *запросов* будут автоматически выделяться.

Поле **ТекстЗапроса** будет содержать текст *запроса*, который мы можем либо написать вручную, либо создать, воспользовавшись конструктором *запроса*.

Теперь добавим в форму еще один элемент управления - табличное *поле*. Зададим ему имя - **РезультатВыполненияЗапроса**. Так же добавим в форму поясняющие надписи: " **Введите текст запроса** " и " **Результат выполнения запроса** ".

В табличное *поле* мы будем помещать результат выполнения *запроса*.

Теперь зададим обработчик нажатия кнопки **Выполнить**. Для этого откроем окно свойств кнопки и нажмем на кнопку **Открыть** в *поле* **Действие**. Процедура обработчика события нажатия на кнопку будет выглядеть следующим образом:

```
Процедура КнопкаВыполнитьНажатие(Кнопка)
   Запрос = Новый Запрос;
   Запрос.Текст=ЭлементыФормы.ТекстЗапроса.ПолучитьТекст();
   РезультатВыполненияЗапроса = Запрос.Выполнить().Выгрузить();
   ЭлементыФормы.РезультатВыполненияЗапроса.СоздатьКолонки();
КонецПроцедуры
```

Поясним ее команды. Они, в основном, связаны с новым для вас объектом **Запрос**.

Запрос = Новый Запрос;

Создаем новый *объект* типа **Запрос**, записываем ссылку на него в переменную Запрос.

Запрос.Текст=ЭлементыФормы.ТекстЗапроса.ПолучитьТекст();

Записываем в свойство *запроса* **Текст** данные, которые хранятся в *поле* текстового документа.

РезультатВыполненияЗапроса = Запрос.Выполнить().Выгрузить();

Помещаем в *поле* табличного документа **РезультатВыполненияЗапроса** результат выполнения *запроса*. Этот результат получается, во-первых, после использования метода *запроса* Выполнить(). Этот метод выполняет *запрос*, если *запрос* выбрал какие-то данные из базы, возвращаемое *значение* имеет тип **РезультатЗапроса**. Метод Выгрузить() выгружает результат *запроса* в таблицу значений, которая и попадает в *поле* табличного документа **РезультатВыполненияЗапроса**.

ЭлементыФормы.РезультатВыполненияЗапроса.СоздатьКолонки();

Благодаря этой команде в *поле* **РезультатВыполненияЗапроса** можно увидеть его содержимое, иначе оно будет выглядеть пустым.

После того, как создание *обработки* завершено, сохраним ее командой главного *меню* программы **Файл > Сохранить**.

Файлы внешних обработок имеют расширение *.EPF.

Запустим систему в режиме 1С:Предприятие. Для того, чтобы работать с внешней *обработкой*, нам сначала нужно открыть ее. Для этого воспользуемся командой главного *меню* программы **Файл > Открыть** и с помощью стандартного окна открытия файлов выберем интересующую нас *обработку*,.

Сейчас все готово для ваших первых экспериментов с *запросами*.

Пожалуй, самый лучший способ изучения языка *запросов* - это исследование его с помощью конструктора *запросов*, а так же самостоятельная ручная модификация *запросов*, полученных после работы конструктора. Когда к вам придет достаточно уверенное понимание языка *запросов* и его конструкций, от модификации готовых текстов можно перейти к написанию собственных.

Щелкнем правой кнопкой мыши *по* свободному пространству в *поле* ввода текста *запроса* в *обработке* **Консоль запросов**. В появившемся контекстном *меню* выберем *пункт* **Конструктор запроса**. Появится соответствующее окно.

Конструктором *запроса* можно пользоваться и в режиме конфигурирования, причем, там он, преимущественно, и используется при составлении *запросов*.

Окно конструктора *запроса* имеет несколько вкладок. Вкладка **Таблицы и поля** служит для указания таблиц (*реальных* и *виртуальных*), из которых будут отбираться данные, и конкретных полей этих таблиц, которые и попадут в результаты *запроса*.

Вкладка **Таблицы и поля** имеет три области. Область, озаглавленная как **База данных** показывает нам набор таблиц, которые имеются в системе. Можно заметить, что эта область напоминает *дерево конфигурации*. Группы можно разворачивать, тогда мы получаем *доступ* к конкретным таблицам. Как видите, здесь представлена *база данных* конфигурации, над которой мы все это время работаем. Развернем несколько группировок.

Среди таблиц есть *реальные* и *виртуальные*. Например, *реальные таблицы* - это таблицы справочников (**ФизическиеЛица**, **Номенклатура**), документов (**ПоступлениеМатериалов**), регистров накопления (**ОстаткиМатериалов**). *Виртуальные таблицы* в данный момент имеют *отношение* к регистру накопления **ОстаткиМатериалов**. В частности, *виртуальная таблица* **ОстаткиМатериалов.Обороты** предназначена для получения информации об оборотах *по* регистру. *Таблица* **ОстаткиМатериалов.Остатки** - для получения данных об остатках, *таблица* **ОстаткиМатериалов.ОстаткиИОбороты**, как следует из ее названия, предназначена для получения и информации об остатках и информации об оборотах.

Рабочая область **Таблицы** окна **Конструктор запроса** содержит *список* таблиц, из которых система выбирает данные. Этот *список* формируется переносом нужных таблиц из рабочей области **База данных**.

Зададимся следующей целью: нужно сконструировать *запрос*, который выводит *список* фамилий сотрудников, работающих в организации.

Очевидно, что данные для такого *запроса* будут храниться в справочнике **Сотрудники**. Поэтому перенесем таблицу **Сотрудники** из области **База данных** в область **Таблицы**, рис. и развернем таблицу.

Если вспомнить устройство справочника **Сотрудники**, то окажется, что один из его реквизитов имеет тип **СправочникСсылка.ФизическиеЛица**, именно из этого реквизита можно "вытащить" фамилию сотрудника. Развернем группировку **ФизическоеЛицо**, найдем среди появившихся полей *поле* **Фамилия** и перенесем его в рабочую область **Поля**,.

Собственно говоря, *запрос*, который выполняет вышеописанную задачу, готов. Для того, чтобы просмотреть текст полученного *запроса*, не выходя из конструктора *запроса*, можно нажать на кнопку **Запрос**, которая расположена в левом нижнем углу формы конструктора. Нажмем на эту кнопку.

Запрос, который отображается в этом окне, можно отредактировать, нажав на кнопку командной панели **Редактировать запрос**, его можно выполнить и посмотреть на результаты выполнения *запроса*, нажав на кнопку **Выполнить запрос**. *Команда* выполнения *запроса* доступна лишь при вызове конструктора *запроса* в режиме 1С:Предприятие.

Закроем окно предварительного просмотра *запроса* и нажмем на кнопку **ОК** в окне **Конструктор запроса**.

Как вы уже могли понять, в *поле* **ТекстЗапроса** попадет следующий текст:

```
ВЫБРАТЬ
   Сотрудники.ФизическоеЛицо.Фамилия
ИЗ
   Справочник.Сотрудники КАК Сотрудники
```

Весь этот текст представляет собой описание *запроса*. *Ключевое слово* ***ВЫБРАТЬ*** обязательно должно присутствовать в начале описания. Оно показывает, какие именно данные должен получать *запрос*.

Ключевое слово ***ИЗ*** позволяет задавать источники данных для *запроса*. В нашем случае мы видим здесь следующий текст:

```
Справочник.Сотрудники КАК Сотрудники
```

Это *выражение* означает, что источником данных является *таблица* **Справочник.Сотрудники**. Для этой таблицы, с помощью ключевого слова ***КАК*** задан *псевдоним* **Сотрудники**. Именно *по* этому псевдониму в рамках *запроса* осуществляется обращение к таблице **Справочник.Сотрудники**.

Вернемся к пояснениям области *ВЫБРАТЬ*. Здесь имеется следующая *команда*:

```
Сотрудники.ФизическоеЛицо.Фамилия
```

Этой командой мы обращаемся к таблице **ФизическоеЛицо**, которая хранится в таблице **Сотрудники** (напомним, что это - *псевдоним*) и выбираем из нее фамилию сотрудника. Полям, которые попадают в выборку данных, создаваемую *запросом*, так же можно назначать псевдонимы с помощью ключевого слова *КАК*.

Если попытаться выразить текст рассматриваемого *запроса* на естественном языке, то получится следующее. Нужно выбрать из таблицы **Справочник.Сотрудники** (которую в *запросе* будем называть просто **Сотрудники**) фамилию сотрудника, которая хранится в таблице **ФизическоеЛицо**, соответствующей конкретному сотруднику.

Теперь нажмем на кнопку **Выполнить** в форме *обработки* **Консоль запросов**.

Продолжим знакомство с *запросами*. Решим несколько задач, направленных на получение данных из нашей базы. Будем писать *запросы* вручную. К **Конструктору запросов** мы еще вернемся, сейчас наша задача - сконцентрировать внимание на текстах *запросов*, понять их сущность.

Получим из справочника **ЕдиницыИзмерения** коды и наименования единиц измерения. В результатах *запроса поле* **Код** должно иметь наименование " **Код** ", *поле* " **Наименование** " - " **Название** ". Справочник **ЕдиницыИзмерения** нужно представить в *запросе* как **ТабЕдИзм**.

Написание *запроса* удобно начинать с группы **ИЗ**. Напишем следующий *шаблон*:

```
ВЫБРАТЬ
ИЗ
   Справочник.ЕдиницыИзмерения как ТабЕдИзм
```

После того, как *источник данных* определен, определяем поля для выбора:

```
ВЫБРАТЬ
   ТабЕдИзм.Код,
   ТабЕдИзм.Наименование КАК Название
ИЗ
   Справочник.ЕдиницыИзмерения как ТабЕдИзм
```

Исполним *запрос*.

Усложним задачу. Результат предыдущего *запроса* нужно упорядочить *по* полю **Название**.

Для этого в *запросе* нужно использовать конструкцию **УПОРЯДОЧИТЬ ПО**. Теперь наш *запрос* будет выглядеть так:

```
ВЫБРАТЬ
   ТабЕдИзм.Код,
   ТабЕдИзм.Наименование КАК Название
ИЗ
   Справочник.ЕдиницыИзмерения как ТабЕдИзм
УПОРЯДОЧИТЬ ПО
   Название
```

При составлении *запроса* можно задавать выбор не всех результатов, полученных из базы, а лишь тех, которые удовлетворяют какому-либо условию. Наиболее простое проявление этого утверждения - выбор некоторого количества первых элементов результата. Реализуется это с использованием слова **ПЕРВЫЕ** с указанием количества отбираемых элементов. Обратите внимание на то, что элементы выбираются с учетом их упорядочивания, заданного в *запросе*.

Выберем первые 2 элемента из результатов, модифицировав предыдущий *запрос*:

```
ВЫБРАТЬ ПЕРВЫЕ 2
   ТабЕдИзм.Код,
   ТабЕдИзм.Наименование КАК Название
ИЗ
   Справочник.ЕдиницыИзмерения как ТабЕдИзм
УПОРЯДОЧИТЬ ПО
   Название
```

Выполним *запрос*.

Для того чтобы в выборку попали лишь различные элементы, можно воспользоваться ключевым словом **РАЗЛИЧНЫЕ** после слова *ВЫБРАТЬ*.

Выше мы делали выборки из неиерархического справочника **ЕдиницыИзмерения**. Попробуем теперь поработать с *иерархическим справочником*. Выберем все элементы из справочника **Номенклатура**. Для выбора всех полей можно воспользоваться командой "*", рис. 7.14.

```
ВЫБРАТЬ
   *
ИЗ
   Справочник.Номенклатура
```

Как видите, в результат *запроса* попали не только элементы справочника (**Одеколон**, **Стрижка** и т.д.), но и группы (**Парфюмерия**, **Парикмахерские услуги**). Для того, чтобы при работе с *иерархическим справочником* ограничить выбор всеми элементами, исключая группы, в текст *запроса* нужно ввести

условие отбора. Обратите внимание на *поле* **ЭтоГруппа**, его можно увидеть на рис. 7.14. Это *поле* у обычных элементов справочника принимает *значение ложь*, у групп - *истина*.

Для отбора элементов можно использовать *ключевое слово* ***ГДЕ***. Оно позволяет задавать условия отбора. Напишем *запрос*, который отбирает наименования элементов справочника **Номенклатура**, исключая наименования групп.

```
ВЫБРАТЬ
   ТабНоменклатура.Наименование
ИЗ
   Справочник.Номенклатура КАК ТабНоменклатура
ГДЕ
   НЕ ТабНоменклатура.ЭтоГруппа
```

Рассмотрим еще один вариант отбора результатов *запроса*, а заодно еще немного продвинемся в изучении **Конструктора запроса**.

Нам нужно отобрать среди документов поступления материалов лишь те, сумма которых превышает 2000 рублей, и вывести в результаты *запроса* даты, номера документов, ФИО материально-ответственного лица, принявшего материалы, а так же сумму документа, упорядочив выводимые строки *по* сумме документа.

Откроем **Конструктор запроса** и на вкладке **Таблицы и поля** отберем в область **Таблицы** таблицу документов **ПоступлениеМатериалов**, в область **Поля** перенесем следующие поля:

- **ПоступлениеМатериалов.Дата**
- **ПоступлениеМатериалов.Номер**
- **ПоступлениеМатериалов.ОтветственныйСотрудник**
- **ПоступлениеМатериалов.СтоимостьМатериалов**

Теперь нам нужно задать условие отбора. Для этого перейдем на вкладку **Условия**. Здесь нам нужно перенести *поле* **СтоимостьМатериалов** из области **Поля** в область задания условий. После этого нужно выбрать условие выбором их списка (мы выбрали условие "больше") и заменить *выражение* **СтоимостьМатериалов** на число 2000.

Здесь нас подстерегает одна особенность конструктора *запросов*. Дело в том, что и *выражение* **СтоимостьМатериалов**, и число 2000 он будет воспринимать как *имя переменной*, которая должна быть передана в *запрос* до его исполнения. Нам же нужно, чтобы 2000 было воспринято именно как число, а не как *имя переменной*. Для того, чтобы это сделать, установим галочку в *поле* **Произвольное**. Это приведет к тому, что *поле* **Условие** превратится в одну строку, в которой будет написано следующее:

```
ПоступлениеМатериалов.СтоимостьМатериалов > &2000
```

Символ **&** нам не нужен - он, как раз, и даст понять системе, что 2000 - это *переменная*. Поэтому мы удалим его вручную, и, в итоге, получим следующий вид вкладки **Условие**.

Для того чтобы упорядочить результат *запроса по* сумме документа, перейдем на вкладку **Порядок**, из области **Поля** добавим в область настройки упорядочивания *поле* **СтоимостьМатериалов** и зададим их упорядочивание.

Работа с **Конструктором запроса** завершена. Нажмем на кнопку **ОК**, текст *запроса* попадет в *поле обработки* **Консоль запросов**, нажмем на кнопку **Выполнить** и убедимся в том, что *запрос* выполнен правильно.

Вот, какой текст сгенерировал **Конструктор запроса**:

```
ВЫБРАТЬ
   ПоступлениеМатериалов.Дата,
   ПоступлениеМатериалов.Номер,
   ПоступлениеМатериалов.ОтветственныйСотрудник,
   ПоступлениеМатериалов.СтоимостьМатериалов КАК СтоимостьМатериалов
ИЗ
   Документ.ПоступлениеМатериалов КАК ПоступлениеМатериалов
ГДЕ
   ПоступлениеМатериалов.СтоимостьМатериалов > 2000

УПОРЯДОЧИТЬ ПО
   СтоимостьМатериалов
```

Выше мы занимались получением данных из какой-то одной таблицы. Существует и другой вариант получения данных - с использованием нескольких таблиц. В целом порядок работы при использовании в качестве источников нескольких таблиц, как правило, взаимосвязанных *по* каким-то признакам, очень похож на работу с отдельными таблицами. Однако если *запрос* выбирает данные из нескольких таблиц, нужно настроить взаимосвязь этих таблиц, а настройка такой взаимосвязи требует, для начала, получения дополнительных знаний о том, как соединяются результаты *запросов по* разным таблицам.

Существуют следующие виды соединения таблиц:

1. *Левое внешнее соединение*;

2. Правое *внешнее соединение*;
3. *Полное внешнее соединение*;
4. Внутреннее соединение;

Рассмотрим эти виды соединения таблиц на примере.

В нашей конфигурации есть два взаимосвязанных справочника. А именно, речь идет о справочниках **Номенклатура** и **ЕдиницыИзмерения**. Элементам справочника **Номенклатура** могут быть назначены единицы измерения, взятые из справочника **ЕдиницыИзмерения**. Причем, в справочнике **Номенклатура** могут существовать элементы, которым не назначена ни одна *единица измерения*, а в справочнике **ЕдиницыИзмерения** могут присутствовать элементы, которые не назначены элементам справочника **Номенклатура**. В таблице 7.1 представлена *таблица* справочника **Номенклатура** - наименования номенклатурных позиций и соответствующие им единицы измерения.

Таблица 7.1. Таблица справочника Номенклатура

Номенклатура	**Единица измерения**	**Группа**
Бальзам для волос		ложь
Духи	Штука	ложь
Завивка	Час	ложь
Лак для волос	Упаковка	ложь
Одеколон	Упаковка	ложь
Парикмахерские услуги		истина
Парфюмерия		истина
Стрижка	Час	ложь
Уход за волосами		истина

Как видите, **Бальзам для волос** не имеет установленной единицы измерения, остальным элементам единицы измерения назначены. Группам, естественно, единицы измерения так же не назначены.

В таблице 7.2 вы можете видеть состав справочника **ЕдиницыИзмерения**.

Таблица 7.2. Таблица справочника ЕдиницыИзмерения

Единица измерения
Килограмм
Метр
Упаковка
Час
Штука

Выводы.

В процессе данной лабораторной работы мы научились формировать, конфигурировать и выполнять запросы разного типа.

3 Методические указания для организации самостоятельной работы

3.1 Общие положения

Целями самостоятельной работы являются систематизация, расширение и закрепление теоретических знаний, приобретение навыков исследовательской деятельности.

Самостоятельная работа студента по дисциплине «Дискретная математика» включает следующие виды его активности:

1. проработка лекционного материала;
2. подготовка к плабораторным занятиям;
3. подготовка к зачету.

3.2 Проработка лекционного материала

Данный вид самостоятельной работы направлен на получение навыков работы с конспектом, структурирования материала, а также умения выделить основные пункты и положения, изложенные на

лекции. Кроме того, проработка лекционного материала способствует более глубокому пониманию и прочному запоминанию теоретической части дисциплины.

При проработке лекционного материала необходимо:

1. отработать прослушанную лекцию, то есть прочитать конспект, прочитать учебник и сопоставить его материал с конспектом; восполнить пробелы, если они остались после лекции в силу того, что студент что-то не понял или не успел записать;

2. перед каждой последующей лекцией прочитать предыдущую, чтобы не тратилось много времени для восстановления контекста изучения дисциплины при продолжающейся теме, а также чтобы максимально правильно ответить на вопросы теста, который проводится на каждой лекции.

Для наиболее эффективной работы с конспектом рекомендуется сначала просмотреть его целиком, чтобы выделить структуру лекции. Эту структуру полезно выписать в виде плана. Затем по каждому пункту нужно выделить основные положения, определения и формулы, если они есть. Формулы тоже полезно записывать, чтобы кроме зрительной, включалась еще и моторная память.

3.3 Подготовка к лабораторным занятиям

Для подготовки к лабораторным занятиям необходимо изучить теоретические вопросы по теме работы, проработать основные понятия, необходимые для решения практических задач и выполнения индивидуального задания по практической работе.

4. Рекомендуемая литература

1. Предметно-ориентированные экономические информационные системы: Учебное пособие / А. И. Исакова - 2016. 239 с. [Электронный ресурс]: — Режим доступа: https://edu.tusur.ru/publications/6542. 7.

2. Голубева, О. Л. 1с [Электронный ресурс]: бухгалтерия : учебник для вузов / О. Л. Голубева. — Москва : Издательство Юрайт, 2021. — 158 с. — (Высшее образование). — ISBN 978-5- 534-14685-1. [Электронный ресурс]: — Режим доступа: https://urait.ru/bcode/479049. 7.