САМОУЧИТЕЛЬ

bhv

1С:Предприятие 7.7

Уроки программирования

Сергей Постовалов
Анастасия Постовалова

- Администрирование и конфигурирование
- Бухгалтерский и оперативный учет
- Расчет заработной платы
- Экспорт и импорт данных
- Подготовка к аттестационным экзаменам

Эффективное обучение
до уровня аттестационного экзамена фирмы «1С»

Сергей Постовалов
Анастасия Постовалова

1С:Предприятие 7.7
Уроки программирования

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.2
П63

Постовалов С. Н., Постовалова А. Ю.

П63 1С:Предприятие 7.7. Уроки программирования. Самоучитель. — СПб.: БХВ-Петербург, 2006. — 320 с.: ил.

ISBN 978-5-94157-777-4

Описываются администрирование системы 1С:Предприятие 7.7, введение в бухгалтерский учет, встроенный язык и основные базовые объекты системы. Рассматривается специфика работы с объектами компонент "Бухгалтерский учет", "Оперативный учет" и "Расчет", а также механизмов экспорта-импорта данных. Обучение проводится на примерах решения стандартных задач, возникающих как при настройке типовых конфигураций в системе 1С:Предприятие 7.7, так и при разработке конфигураций по ведению бухгалтерского и оперативного учета, расчета заработной платы. По каждой теме даны задания и контрольные вопросы. В конце книги приведены ответы и решения наиболее сложных задач.

Книга предназначена как для программистов, никогда раньше не работавших с 1С:Предприятием 7.7, так и для опытных специалистов по 1С. Ее можно использовать для подготовки к аттестационным экзаменам по программам 1С:Бухгалтерия 7.7, 1С:Торговля и Склад 7.7, 1С:Зарплата и Кадры 7.7.

Для 1С-программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Татьяна Лапина*
Компьютерная верстка	*Татьяны Олоновой*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.01.06.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 25,8.
Тираж 3000 экз. Заказ № 47
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-94157-777-4

© Постовалов С. Н., Постовалова А. Ю., 2006
© Оформление, издательство "БХВ-Петербург", 2006

Оглавление

Введение

Книга, которую вы держите в руках, является учебным пособием для самостоятельного изучения программирования в системе 1С:Предприятие 7.7. Если вас заинтересовала данная тема, значит вы, уважаемый читатель, уже слышали про эту замечательную программу. Может быть, кто-то улыбнется, а кто-то захочет заключить слово "замечательную" в кавычки, но не в этом суть. Рынок в лице массового потребителя выбрал этот продукт в качестве основного инструмента для автоматизации учета и, тем самым, создал огромный спрос на специалистов по 1С.

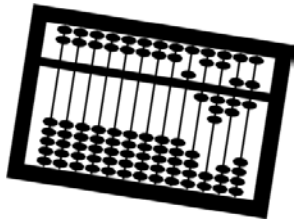
Когда задают вопрос "Сложно ли научиться программировать на 1С?", мы отвечаем: "И просто, и сложно". Просто — потому что очень легко сделать первый шаг человеку, немного знакомому с программированием на любом другом языке, например Паскале или Бейсике. Сложно — потому что нужно знать и уметь использовать ряд специфических механизмов, реализованных в системе 1С:Предприятие 7.7, чтобы *эффективно* программировать на 1С. Даже многолетний опыт *успешной* работы специалиста по 1С еще не гарантирует знания всех тонкостей работы механизмов системы.

Материал книги можно разделить на две части. "Первый шаг" — *главы* с *1* по *6*, которые описывают администрирование системы, введение в бухгалтерский учет, описание встроенного языка и основные базовые объекты системы 1С:Предприятие 7.7. "Второй шаг" — это *главы* с *7* по *12*, описывающие специфику работы с объектами компонент "Бухгалтерский учет", "Оперативный учет" и "Расчет", а также механизмов экспорта-импорта данных.

Материал дается "с нуля", т. е. исходя из предположения, что вы никогда не видели систему 1С:Предприятие 7.7. Однако даже если вы ежедневно работаете с этой системой, книга может пригодиться для получения цельного представления о ней и для подготовки к аттестационным экзаменам, проводимым фирмой "1С".

Чтение книги должно чередоваться с выполнением практических заданий на компьютере. Только закрепив материал таким образом, есть смысл читать дальше, поскольку обучение построено по принципу "от простого — к сложному". В конце каждой главы приводится список контрольных вопросов. Сразу предупредим, что не на все вопросы ответ дан именно в этой главе. Над чем-то нужно подумать, что-то посмотреть в синтакс-помощнике или документации к системе 1С:Предприятие 7.7, что-то проверить экспериментально на компьютере. Решения и ответы на наиболее сложные вопросы и задания приведены в конце книги.

Ну, а теперь пора начинать!



Глава 1

Введение в систему 1С:Предприятие 7.7

Даже бегло ознакомившись с объявлениями о существующих вакансиях на должность бухгалтера, нетрудно заметить, что одним из главных требований к кандидатам является знание "1С". Если же просмотреть вакансии программистов, станет ясно, что существенную долю на данном рынке занимают программисты "1С". Что это за два волшебных знака, которые прочно закрепились в нашем лексиконе? В данной главе мы познакомимся с историей создания системы 1С:Предприятие 7.7, входящими в нее компонентами, и основами ее администрирования.

1.1. О фирме "1С"

Фирма "1С" специализируется на дистрибуции, поддержке и разработке компьютерных программ и баз данных делового и домашнего назначения. Основанная в 1991 году, "1С" — российская фирма со штатом более 400 человек, опирающаяся исключительно на собственные профессиональные разработки. По данным многочисленных опросов, "1С" занимает первое место в софтверном секторе российской компьютерной индустрии (**http://www.1c.ru**).

Впрочем, даже если бы опросы не проводились, следует признать — первенство "1С" в настоящий момент очевидно.

Обратимся к истории. В 1992 году на выставке "Comtek" демонстрируется первая бухгалтерская программа фирмы "1С". Называлась она "Мини-Бухгалтерия", была простой и удобной в работе, и в результате грамотной маркетинговой политики быстро вышла в лидеры продаж среди бухгалтерских программ. Через два года выходит 1С:Бухгалтерия 6.0 — одна из первых бухгалтерских программ, работающих под Windows. В 1996 году появляется продукт нового поколения —1С:Торговля 7.0 — первый кирпичик на новой технологической платформе, которая окончательно оформляется в 1998 году под названием 1С:Предприятие 7.7. За прошедшее время пользователями программ для управления и учета системы 1С:Предприятие 7.7 стало более 700 000 организаций. Однако фирма "1С" не привыкла жить на старых заслу-

гах и практически сразу приступила к разработке следующей версии системы 1С:Предприятие 8.0, которая появилась в 2003 году. Наверное, у читателя может возникнуть закономерный вопрос: "А стоит ли изучать версию 7.7, если уже есть новая версия? И сильно ли они отличаются?"

Прежде всего, отметим, что 1С:Предприятие 7.7 и 1С:Предприятие 8.0 — это две *разные* программы, схожие на уровне концепций. Обе версии позволяют реализовать практически любую учетную задачу. Восьмая версия обладает более развитым инструментарием для разработчика и обладает большей масштабируемостью, чем версия 7.7. В то же время, седьмая версия менее требовательна к аппаратным ресурсам и более простая в освоении. Поэтому, скорее всего, седьмая и восьмая версии 1С:Предприятия будут достаточно долгое время существовать параллельно.

В деятельности фирмы "1С" можно выделить несколько ключевых моментов. Прежде всего, *индустриальный подход к разработке, тиражированию, продаже и поддержке программ*. Что это значит? Основной продукт фирмы "1С" — система 1С:Предприятие 7.7 — является "конструктором" для создания мощных программ автоматизации предприятия малыми силами. На базе системы 1С:Предприятие 7.7 фирмой "1С" разработаны типовые решения в области бухгалтерского учета, оперативного (управленческого) учета и расчета заработной платы. Очевидно, что каждая отрасль имеет свои особенности, и охватить все области фирма "1С" не в состоянии, поэтому разработкой отраслевых решений занимаются ее партнеры.

Второй ключевой момент — *опора на широкую партнерскую сеть, обеспечивающую качественное и эффективное обслуживание массового потребителя и высокие темпы развития*. В настоящее время сеть насчитывает больше трех тысяч фирм-франчайзи. Из них около 200 занимаются разработкой собственных конфигураций, ориентированных на отраслевые решения, такие как торговля, транспорт, туризм, услуги и т. д.

Не менее важным моментом является то, что *создана система обучения и аттестации специалистов по 1С*. Качество специалистов подтверждается наличием сертификатов, которые выдаются после сдачи аттестационных экзаменов. Экзамены проводятся как в учебных центрах фирмы "1С", так и в крупных городах специальными выездными комиссиями. Более подробную информацию об этом можно получить на сайте **http://www.1c.ru** в разделе "Обучение и аттестация". Там же можно найти и примеры экзаменационных задач. Решения некоторых из этих задач мы будем рассматривать в настоящей книге.

1.2. О системе 1С:Предприятие 7.7

1С:Предприятие 7.7 является гибкой настраиваемой системой для решения широкого круга задач в сфере автоматизации деятельности предприятий.

Программы, в которые программист может вносить изменения, обычно называются программами с открытым кодом. 1С:Предприятие 7.7 можно частично отнести к программам с открытым кодом, то есть часть кода является открытой, а часть — закрытой. Закрытая часть — это "платформа" или "ядро", открытая — это "конфигурация". На рис. 1.1 приведена схема взаимодействия различных компонент 1С:Предприятия. В системе 1С:Предприятие 7.7 можно выделить три уровня:

- технологическая платформа;
- прикладные компоненты;
- конфигурации.

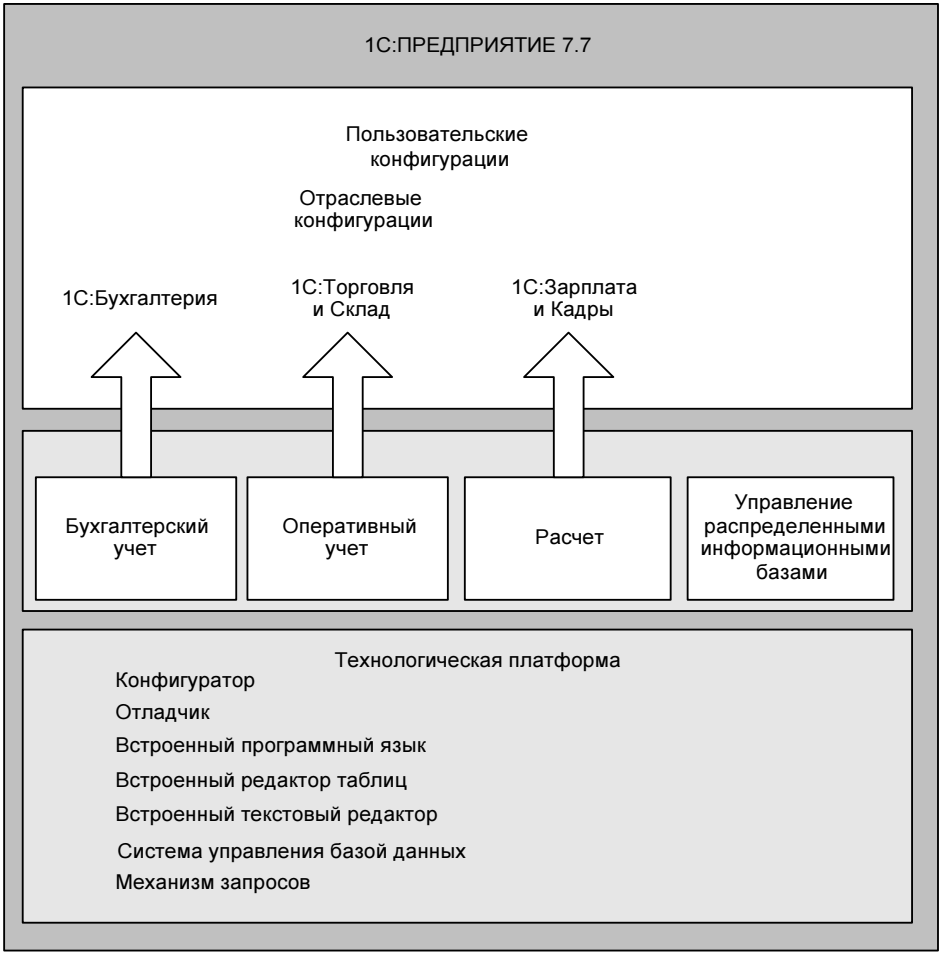


Рис. 1.1. Структурная схема системы 1С:Предприятие 7.7

1.2.1. Технологическая платформа

Технологическая платформа включает в себя конфигуратор, отладчик, монитор, встроенный программный язык, встроенный редактор таблиц, встроенный текстовый редактор, систему управления базой данных и механизм запросов.

Конфигуратор — это программа для конфигурирования и администрирования системы. *Конфигурирование* — это определение структуры данных информационной базы, разработка форм диалогов, описание алгоритмов функционирования системы на встроенном языке программирования, настройка наборов прав и интерфейсов пользователей. *Администрирование* — это управление списком пользователей, сохранение резервных копий информационной базы, тестирование и исправление информационной базы, настройка журнала регистрации.

Отладчик — это программа для выполнения программных модулей в пошаговом режиме. Также в отладчике можно вычислять текущие значения переменных, выражения и функции. Полезной возможностью отладчика является замер производительности. С момента включения замера производительности и до его выключения отладчик запоминает количество вызовов операторов и процедур и время выполнения каждой операции. Таким образом, можно точно определить, на что больше всего времени тратится при выполнении программы.

Монитор — это программа для просмотра списка активных пользователей, работающих с информационной базой, и журнала регистрации действий пользователя, который хранится в файле 1cv7.mlg.

Система управления базой данных (СУБД) позволяет разработчику абстрагироваться от физического уровня хранения данных. Достигается это за счет того, что структура данных описывается на языке предметной области — настраиваются реквизиты документов, справочников, операций и проводок. При сохранении *конфигурации* система генерирует набор файлов в формате DBF в файл-серверной версии программы или набор таблиц в базе данных под управлением MS SQL Server. В дальнейшем разработчик оперирует данными не на уровне записей таблиц, а на уровне *объектов* предметной области, причем информация об одном объекте фактически может храниться в нескольких физических таблицах. Для извлечения информации может использоваться универсальный механизм запросов, *см. главу 5.*

1.2.2. Прикладные компоненты

Каждая прикладная компонента содержит набор объектов специализированных для определенной предметной области. Компонента ***"Бухгалтерский учет"*** содержит объекты "Планы Счетов", "Виды Субконто", "Операция", "Бухгалтерские Итоги". Компонента ***"Оперативный учет"*** содержит объект

"Регистры". Компонента ***"Расчет"*** содержит объекты "Журналы Расчетов", "Виды Расчетов", "Группы Расчетов", "Календари". Специальная компонента ***"Управление распределенными информационными базами"*** позволяет производить обмен информацией между удаленными рабочими местами.

1.2.3. Конфигурации

Фирма "1С" не продает отдельно платформу 1С:Предприятие 7.7. На базе этой платформы разработаны и поддерживаются типовые конфигурации, которые и поставляются с набором необходимых компонент. В табл. 1.1 перечислены основные типовые конфигурации.

Таблица 1.1. *Типовые конфигурации*

Типовая конфигурация	Бухгалтерский учет	Оперативный учет	Расчет
"Бухгалтерский учет"	+		
"Торговля и Склад"		+	
"Зарплата и Кадры"			+
"Производство+Услуги+Бухгалтерия"	+	+	
Комплексная "Бухгалтерия+ Торговля+Склад+Зарплата+Кадры"	+	+	+
"Платежные документы"			
"Деньги"		+	
"Финансовое планирование"		+	

Одной из характерных особенностей конфигураций является их масштабируемость: одна и та же конфигурация может работать на локальном компьютере, в сети и на сервере под управлением MS SQL Server.

1.3. Система защиты системы 1С:Предприятие 7.7

Программа 1С:Предприятие 7.7 защищена аппаратным ключом. В настоящее время выпускаются ключи двух видов — для LPT- и для USB-портов компьютера. Отличием сетевых ключей является их цвет — красный. Перед запуском системы 1С:Предприятие 7.7 необходимо установить драйвер защиты. Желательно устанавливать самую последнюю версию драйвера защиты. Ее

можно взять с дисков информационно-технологического сопровождения (ИТС) или скачать с сайта производителя ключей **http://www.aladdin.ru**.

При использовании сетевой версии на компьютере, к которому присоединен аппаратный ключ, устанавливается *сервер защиты*. Способ установки сервера зависит от используемой операционной системы и описан в руководстве по инсталляции программы. Есть возможность устанавливать ключ и сервер защиты на компьютере с операционной системой Linux.

1.4. Запуск системы 1С:Предприятие 7.7

При запуске системы 1С:Предприятие 7.7 появляется диалоговая форма (рис. 1.2), в которой производится выбор информационной базы и режима работы ("Предприятие", "Конфигуратор", "Отладчик", "Монитор").

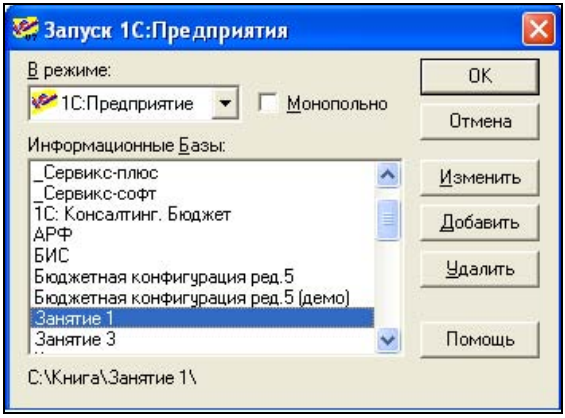


Рис. 1.2. Запуск системы 1С:Предприятие 7.7

1.4.1. Параметры командной строки

Можно ли запустить 1С:Предприятие 7.7 без открытия окна выбора информационной базы данных? Да, можно, для этого надо указать параметры командной строки:

- CONFIG — конфигуратор;
- DEBUG — отладчик;
- ENTERPRISE — предприятие;
- /D<Путь> — каталог информационной базы данных;
- /U<Путь> — каталог пользователя;

- /N<Имя> — имя пользователя, как оно указанно в списке пользователей в конфигураторе;
- /P<Пароль> — пароль пользователя;
- /M — монопольный режим;
- /T<Путь> — параметр для переопределения каталога временных файлов.

ПРИМЕР

Чтобы запустить информационную базу № 1, расположенную в каталоге `C:\Книга\Занятие 1`, под пользователем `Иванов` с паролем `123`, нужно указать строку запуска: `"C:\Program Files\1cv771\BIN\1cv7.exe" enterprise /D"C:\Книга\Занятие 1" /NИванов /P123`.

1.4.2. Первый запуск

Если у вас сетевая версия 1С:Предприятия 7.7, то первый запуск нужно производить в монопольном режиме, так как происходит создание индексных файлов. Также монопольный режим запуска требуется и в других случаях: при переиндексации базы данных, при сохранении и восстановлении базы данных, при удалении помеченных на удаление объектов, при групповом перепроведении документов.

1.4.3. Создание новой (пустой) конфигурации

Чтобы создать новую (пустую) конфигурацию, необходимо

1. Выбрать режим **Конфигуратор**.
2. Нажать кнопку **Добавить**.
3. Выбрать (или создать новый) каталог, в котором будет находиться информационная база.
4. Нажать кнопку **ОК**.
5. В меню **Конфигурация** выбрать пункт **Открыть конфигурацию**.
6. В меню **Файл** выбрать пункт **Сохранить**.

Далее на все вопросы отвечать утвердительно. В результате в каталоге информационной базы появятся системные файлы конфигурации. Соответствующий файл конфигурации имеет имя 1cv7.md, словарь данных хранится в файле 1cv7.dd (в SQL-версии 1cv7.dds). Используя словарь данных, можно уточнить, в каких файлах (SQL-таблицах) хранятся информационные объекты.

Задание 1.1

1. Создайте новую (пустую) информационную базу данных в новом каталоге.
2. Откройте конфигурацию. В свойствах задачи задайте имя конфигурации ("Занятие № 1"), введите авторов и пароль на конфигурацию.

1.5. Конфигурация

Открыть конфигурацию можно через меню **Конфигурация**, пункт **Открыть конфигурацию**. Окно конфигурации состоит из трех закладок (рис. 1.3):

- **Метаданные**;
- **Интерфейсы**;
- **Права**.

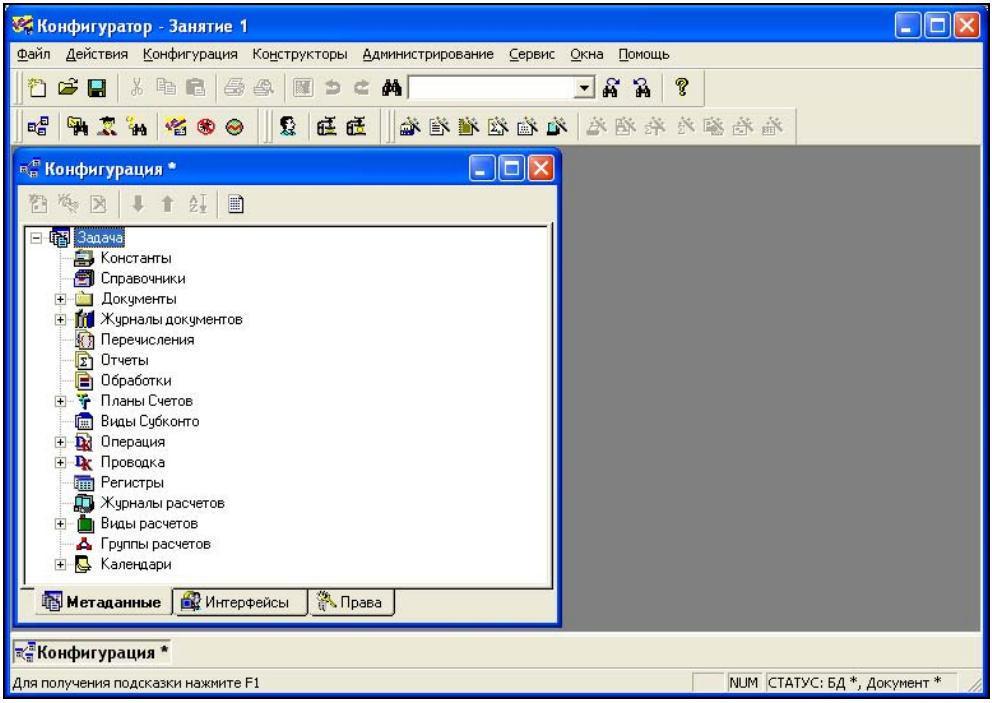


Рис. 1.3. Структура конфигурации

Метаданные — это данные о данных, а именно:

- информация о структуре информационных баз данных: справочников, документов и т. д.;
- формы диалогов и списков;
- таблицы отчетов;
- программные модули, в которых на встроенном языке описываются алгоритмы функционирования системы.

На закладке **Интерфейсы** происходит создание и настройка интерфейсов пользователей. Интерфейс состоит из двух частей: меню и инструменталь-

ных панелей. У интерфейса может быть *подчиненный интерфейс*, который получается из родительского интерфейса перечислением доступных пунктов меню и кнопок.

Внимание!

При настройке интерфейса (пункт **Свойства** контекстного меню) обычного пользователя нужно отключить системное меню **Операции**, через которое можно получить доступ ко всем объектам конфигурации.

На закладке **Права** задаются наборы прав доступа к информационным объектам, определенным в закладке **Метаданные**. При настройке набора прав обратите внимание на права, задаваемые на свойствах задачи. Если поставить флажок **Административные функции**, то пользователь с таким набором прав будет иметь возможность изменять структуру метаданных, управлять списком пользователей, назначать права доступа, редактировать пользовательские интерфейсы.

Задание 1.2

1. Установите текущий релиз типовой конфигурации "Бухгалтерский учет".
2. Создайте в конфигурации новый интерфейс — "ИнтерфейсКассира". Пользователь с таким интерфейсом может вводить приходные и расходные кассовые документы и просматривать список кассовых документов. Настройте панель инструментов и меню.
3. Создайте в конфигурации новый набор прав — "ПраваКассира". Пользователь с такими правами может вводить новые кассовые документы, но не может изменять уже проведенные кассовые документы.

1.6. Ввод пользователей системы

Ввод пользователей системы производится в конфигураторе в меню **Администрирование**, пункт **Пользователи**. Для каждого пользователя задается рабочий каталог, набор прав, интерфейс и пароль (по умолчанию пароля нет). Один из пользователей обязательно должен обладать административными функциями.

Если для пользователя задан рабочий каталог, то при попытке повторного входа под этим пользователем система выдаст сообщение "Каталог пользователя занят".

Список пользователей хранится в фале users.usr в подкаталоге **USERDEF** каталога информационной базы.

Задание 1.3

1. Заведите в конфигураторе двух новых пользователей системы (один кассир, другой — администратор), задайте им пароли, роли и интерфейсы.

2. Запустите конфигурацию под интерфейсом кассира, введите приходный кассовый ордер от покупателя — контрагента Иванова (корреспондирующий счет 62.1) на сумму 1200 рублей. Проверьте возможность изменения непроведенного документа и невозможность изменения проведенного документа.

1.7. Сохранение, восстановление и тестирование информационных баз

В меню **Администрирование** есть пункты для сохранения, восстановления, выгрузки, загрузки и тестирования информационной базы. При сохранении выполняется сжатие баз данных и файла конфигурации в архив с расширением zip. При восстановлении данных происходит обратная операция: из файла архива происходит распаковка баз данных и файла конфигурации. Сохранение и восстановление производят с целью создания резервных копий и для переноса данных с одного компьютера на другой (целиком).

При выгрузке данных информация сначала сохраняется в файле данных, а затем добавляется в архив. С помощью выгрузки-загрузки можно перенести информацию из базы данных в формате DBF в базу данных под управлением MS SQL Server и обратно.

Тестирование и исправление информационных баз производится в случае, когда имеются ошибки в базах данных, связанные, как правило, с системными сбоями (выключение питания, зависание программы и т. д.).

Задание 1.4

1. Сохраните (выгрузите) информационную базу данных.
2. Создайте новую (пустую) информационную базу с названием "Копия".
3. Восстановите (загрузите) в копию информационную базу из архива.
4. Протестируйте информационную базу данных.

1.8. Обновление и загрузка измененной конфигурации

Опыт работы с типовыми конфигурациями показывает, что срок жизни типовой редакции составляет несколько лет. Причем за это время фирма "1С" успевает выпустить несколько десятков релизов. Чем отличается релиз от редакции? Релиз — небольшая модификация конфигурации, связанная с исправлением имеющихся ошибок, выходом новых форм документов и отчетов, небольшими изменениями в законодательстве. Редакция же выпускается, когда меняется методология программы. Это может быть связано как с

существенными изменениями в законодательстве (изменение плана счетов, введение налогового учета), так и потребностью коренных изменений в структуре данных и выполняемых функций. Так, например, самая первая конфигурация — "Торговля и Склад" — пережила 9 редакций; конфигурация "Бухгалтерский учет" — 4; "Зарплата и Кадры" — 2. Количество редакций говорит также о негибкости ранних редакций, что делает невозможным эволюционное развитие программы. Старые редакции фирма "1С" не поддерживает, что приводит к их постепенному отмиранию.

Какие же механизмы дает "1С" для обновления программ?

При переходе от релиза к релизу применяется два способа — **Загрузить измененную конфигурацию** и **Объединение конфигураций**.

- Первый способ — **Загрузить измененную конфигурацию** — применяется, если последующая конфигурация является потомком изменяемой. Последнее означает, что данная конфигурация была скопирована, изменена в другом месте и загружена обратно. Если же после копирования обе конфигурации подверглись изменениям (даже несущественным), то при попытке загрузить другую конфигурацию система выдаст ошибку "Выбранный файл конфигурации не является потомком данного файла. При реструктуризации может произойти разрушение данных". Продолжайте загрузку, только если абсолютно уверены, что изменения, сделанные в текущей конфигурации после копирования, являются несущественными и могут быть потеряны.
- Второй способ — **Объединение конфигураций** — применяется, если требуется объединить две разные конфигурации (когда-то бывшие одной). При этом можно отдать приоритет либо текущей, либо загружаемой конфигурации и выбрать режим замещения или объединения объектов. В режиме замещения объект приоритетной конфигурации замещает объект с таким же именем другой конфигурации. В режиме объединения система пытается объединить два объекта с одинаковыми именами. При объединении конфигураций можно флажками отметить, какие объекты требуется объединить.

  Второй способ работает существенно медленней и, вообще говоря, не всегда заканчивается успешно. Однако именно второй способ позволяет получить подробный отчет о различиях между конфигурациями.

Совет

Если вы не знаете, какие были сделаны изменения в данной конфигурации по сравнению с типовой, можно выполнить объединение с типовой конфигурацией (обязательно нужен тот же самый релиз, что стоит в поле **Комментарий** в свойствах задачи!) и сформировать подробный отчет об изменениях.

Есть еще одна возможность переноса объектов внутри одной конфигурации или между двумя разными — использование буфера обмена. Объект копируется в одном месте, а затем вставляется в другом.

При переходе от редакции к редакции применяются специальные конверторы, которые осуществляют перенос данных между двумя информационными базами. В настоящее время эти конверторы применяют для переноса данных промежуточный файл в формате XML.

Задание 1.5

Запустите типовую конфигурацию и выполните ее объединение с измененной. Какие объекты типовой конфигурации оказались измененными?

1.9. Внесение изменений в типовую конфигурацию

Получив задание пользователя системы по настройке конфигурации, не торопитесь сразу же менять ее программный код. Как правило, многие проблемы у пользователя возникают из-за незнания всех возможностей типовой конфигурации. Возможно, изменения, которые просит сделать пользователь, противоречат законодательству или решаются другими способами. Выясните также, какой объем трудозатрат пользователя экономит программирование данной задачи, и не является ли перепрограммирование типовой конфигурации более сложной задачей (по времени и стоимости).

Если все-таки пользователь настаивает на внесении изменений, то сформулируйте эти изменения в письменном виде, тогда будет проще и сделать, и сдать работу. Внимательно прочитайте техническое задание и попытайтесь его детализировать: если вам написали проводки документа, уточните аналитику по проводкам, если попросили добавить документ, уточните реквизиты документа, возможные движения и проводки, печатную форму документа.

Перед изменением конфигурации нужно выполнить резервное копирование (предварительно все пользователи должны выйти из программы), причем в имени архива желательно указать дату и время архивирования.

В сетевой версии программы одновременно работают несколько пользователей. Чтобы не "выгонять" их из программы при очередном сохранении конфигурации (оно выполняется в монопольном режиме), изменение конфигурации и тестирование сделанных изменений нужно выполнять в копии информационной базы. Для получения точной копии создайте пустую информационную базу (например, в каталоге **Для программирования**), как показано в *разд. 1.4.3 данной главы*, и сделайте восстановление информационной базы из резервной копии.

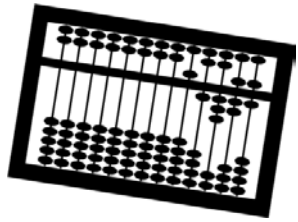
Все изменения надо тщательно документировать. Изменяемый код не удаляется, а комментируется, в комментариях ставится дата изменения, кем сделано изменение, с какой целью. Изменения в структуре данных можно

фиксировать в обработке, специально созданной для этого. В качестве образца можно посмотреть обработку "ОбновлениеИБ" в типовых конфигурациях фирмы "1С". Кроме того, можно в идентификаторах добавляемых объектов и реквизитов использовать свой префикс, например, `спНовыйОбъект`, `спНовыйРеквизит`. Работоспособность сделанных изменений должна быть проверена пользователем, так как только конечный пользователь может оценить правильность вашего решения. Только после этого делаем перенос конфигурации в рабочую базу данных.

Снова заходим в рабочую базу данных и опять делаем резервное копирование с указанием даты и времени (это, кстати, позволит определить время, затраченное на программирование) на случай "незамеченных" ошибок. И только после этого делаем загрузку измененной конфигурации из каталога **Для программирования** и сохраняем сделанные изменения.

1.10. Контрольные вопросы

1. В каком формате могут храниться данные информационной базы?
2. Может ли одна и та же конфигурация работать на локальной и на сетевой версии программы 1С:Предприятие 7.7?
3. Можно ли по цвету *ключа защиты* определить, для какой (локальной или сетевой) версии он предназначен?
4. Как запустить 1С:Предприятие 7.7 без открытия окна диалога выбора информационной базы данных?
5. Когда и зачем выполняется тестирование и исправление информационных баз?
6. Что произойдет, если удалить файл users.usr в подкаталоге **USERDEF** каталога информационной базы?
7. Как узнать, какие были сделаны изменения в данной конфигурации по сравнению с типовой?



Глава 2

Введение в бухгалтерский учет

Система 1С:Предприятие 7.7 позволяет разрабатывать прикладные решения для различных учетных задач, но исторически сложилось так, что первым и основным объектом автоматизации является бухгалтерский учет. В данной главе мы познакомимся с основными понятиями бухгалтерского учета, терминологией, чтобы при разговоре с потенциальным заказчиком-бухгалтером вы могли говорить с ним "на одном языке".

2.1. Бухгалтерский учет, его объекты и основные задачи

Бухгалтерский учет имеет давнюю историю, множество различных форм и теорий. Однако в настоящее время есть единые правила ведения бухгалтерского учета в рамках одного государства, законодательно закрепленные (федеральный закон Российской Федерации № 129-ФЗ). Зачем необходимы единые правила? Для того чтобы данные бухгалтерского учета различных организаций были сопоставимы. Следующий шаг — приведение бухгалтерского учета к международным стандартам финансовой отчетности (МСФО) — готовится в настоящее время. В 1998 году правительством Российской Федерации была утверждена программа реформирования бухгалтерского учета в соответствии с МСФО.

Определение

Бухгалтерский учет представляет собой упорядоченную систему сбора, регистрации и обобщения информации в денежном выражении об имуществе, обязательствах организаций и их движении путем сплошного, непрерывного и документального учета всех хозяйственных операций.

Объектами бухгалтерского учета организации являются:

- имущество:
  - основные средства (средства труда при производстве продукции, выполнении работ или оказании услуг в течение периода, превышающего 12 месяцев);

- нематериальные активы (авторские права, патенты на изобретения, товарные знаки и т. д.);
- материалы (сырье, топливо, запасные части и другие материальные ресурсы, используемые при производстве продукции, выполнении работ или оказании услуг);
- товары (предметы, приобретенные с целью перепродажи);
- продукция (предметы — результат собственного производства);
- денежные средства (наличные и безналичные);
- ценные бумаги;
- другое имущество;

❐ обязательства, возникающие при расчетах:

- с поставщиками товаров и услуг (поставщик берет на себя обязательства поставить товар или оказать услуги, а мы обязуемся их оплатить);
- с покупателями товаров и услуг (мы берем на себя обязательства поставить товар или оказать услуги, а покупатель обязуется их оплатить);
- с сотрудниками (сотрудники берут на себя обязательства трудиться, а мы берем на себя обязательства выплачивать заработную плату);
- с государством (мы берем на себя обязательства рассчитывать и перечислять налоги и взносы в фонды);

❐ хозяйственные операции.

2.2. Основные требования к ведению бухгалтерского учета

К ведению бухгалтерского учета предъявляются следующие требования.

❐ Ведение бухгалтерского учета возможно только на основании *первичных учетных документов*, оформляемых при проведении хозяйственных операций. Эти документы должны составляться по унифицированным стандартным формам, утвержденным Госкомстатом РФ (при их отсутствии разрабатываться самой организацией), и иметь ряд *обязательных реквизитов*: наименование документа, дату составления, наименование организации, содержание хозяйственной операции, измерители хозяйственной операции в натуральном и денежном выражении, наименование должностей ответственных лиц и их личные подписи.

❐ При ведении бухгалтерского учета допускается использование только стандартного плана счетов бухгалтерского учета (на его основе организация может сформировать свой рабочий план счетов). Стандартный

план бухгалтерского учета формируется путем перечисления основных объектов бухгалтерского учета и присвоения им кодов. Например, основные средства — код 01, материалы — код 10, товары — код 41, продукция — код 43 и т. д. Отметим, что в данном случае приведена кодировка плана счетов хозрасчетной организации. Есть свой план счетов для бюджетных и свой — для кредитных организаций.

- Данные бухгалтерского учета и отчетности проверяются и подтверждаются документально путем проведения обязательной *инвентаризации* (сверки учетного остатка и фактического наличия объектов учета). Учетный остаток образуется путем регистрации движений первичных документов по счетам бухгалтерского учета. Однако в документ может вкрасться ошибка (написали одно, выдали другое), также ошибка может быть допущена при записи движений документа (например, при вводе документа в компьютер), наконец, объект может быть похищен или утерян. После инвентаризации составляются акты списания или оприходования, которые выравнивают учетный и фактический остаток.
- Бухгалтерский учет ведется в рублях. Записи по валютным счетам и операциям в иностранной валюте производятся в рублях с пересчетом по курсу Центрального Банка РФ на дату совершения операции. Одновременно эти записи производятся в валюте расчетов и платежей.

2.3. Пример бухгалтерского учета в организации оптовой торговли

Рассмотрим бухгалтерский учет на примере хозяйственных операций в оптовой торговой организации (рис. 2.1).

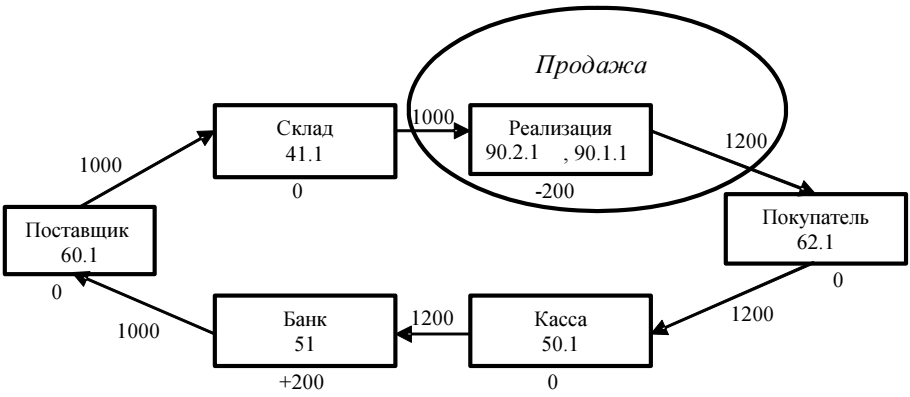


Рис. 2.1. Пример хозяйственных операций

Предположим, мы купили у поставщика 100 дискет по цене 10 рублей на общую сумму 1000 рублей. Затем мы отправили дискеты в магазин и продали покупателям по цене 12 рублей. Покупатели внесли в кассу сумму 1200 рублей. Деньги из кассы мы сдали в банк и рассчитались с поставщиком.

Каждый квадратик на рис 2.1 в бухгалтерии называется *счетом* и имеет свой код. Код является стандартным и принадлежит *плану счетов*, который утверждается государственными органами. Так, например, склад имеет код "41.1", касса — "50.1" и т. д. Счет состоит из двух чисел: код счета и код субсчета.

Каждая стрелка на рис. 2.1 в бухгалтерии называется *проводкой* и обозначает факт перемещения со счета на счет. Счет, с которого начинается стрелка, называется *кредитом*, а счет, в который входит стрелка, — *дебетом* проводки.

Одна хозяйственная операция может состоять из одной проводки, например, поступление денег в кассу (Д51/К62.1), а может состоять из двух и более проводок, например, *продажа* — товар списывается со склада по закупочной цене (Д90.2/К41.1), а отпускается покупателю по продажной цене (Д62.1/К90.1).

Если сложить все суммы, входящие на счет, и отнять все суммы, выходящие со счета, то мы получим остаток по счету, который в бухгалтерии называется сальдо.

Все счета делятся на три вида: активные, пассивные и активно-пассивные. На *активных* счетах сальдо всегда положительное (там деньги накапливаются, например, в кассе или на складе) или *дебетовое*. На *пассивных* счетах сальдо всегда отрицательное (например, реализация — продаем мы по большей цене, чем получаем) или *кредитовое*. На *активно-пассивных* счетах сальдо может быть как положительным, так и отрицательным (например, поставщики или покупатели).

Если просуммировать сальдо по всем счетам, то мы получим ноль. Это как раз и есть критерий проверки *бухгалтерского баланса:* сумма остатков по активным счетам равна сумме остатков по пассивным счетам.

Бухгалтерские счета бывают *балансовыми* и *забалансовыми*. На забалансовых счетах учитываются объекты, которые не влияют на состояние нашей организации. Проводка по забалансовому счету может не иметь корреспондирующего счета. Например, поступление комиссионного товара на склад отражается проводкой

```
Д004, Дискета, Склад №1, количество 100, сумма 1000,
```

а продажа проводкой

```
К004, Дискета, Склад №1, количество 100, сумма 1000.
```

Поскольку комиссионный товар является собственностью комитента (комитент — организация, передавшая нам товар на комиссию), то он не попадает на баланс нашей организации, но в то же время его учет вести необходимо. Поэтому учет ведется на забалансовом счете 004.

В нашем примере на рис. 2.1 после выполнения всех хозяйственных операций мы имеем нулевое сальдо по всем счетам, кроме банка и реализации. В банке у нас осталось 200 рублей по дебету, а на счете реализации 200 рублей по кредиту. В результате мы никому ничего не должны, нам тоже никто ничего не должен, на расчетном счете мы имеем 200 рублей чистой прибыли, источник которых — торговая наценка в 200 рублей при реализации товаров.

Счета позволяют видеть картину в консолидированном или *синтетическом* виде, например, сумму долга по всем поставщикам (кредитовое сальдо по счету 60.1) или сумму всех товаров на складе (сальдо по счету 41.1). Однако не менее важно вести учет *в аналитике*, т. е. по каждому поставщику в отдельности, по каждому товару или по каждому складу.

Для этого в системе 1С:Предприятие используется понятие *субконто*, которое фактически означает код аналитического учета. При вводе проводки мы должны указать не только корреспондирующие счета, но и значения субконто, например, при поступлении товаров на склад, по кредиту мы должны указать от кого мы получили товар (контрагент "НЭТА"), а по дебету — какой товар (номенклатура "Дискеты TDK") и на какой склад (место хранения "Основной склад") мы его поместили.

На каждом счете может быть несколько видов субконто (в стандартной версии 1С:Бухгалтерии 7.7 — до трех, в профессиональной — до пяти). Подробнее о настройках счетов будет рассказано в *главе 7*.

Счет называется *закрытым*, если конечное сальдо равно нулю. Чтобы закрыть счет, формируется проводка на сумму остатка по счету. В нашем примере счет 90 является промежуточным и в конце отчетного периода должен быть закрыт. Если по счету 90 имеется кредитовое сальдо, то мы делаем проводку Д90.9 К99 (Прибыли и убытки) на сумму остатка по счету 90.

Задание 2.1

1. Введите операции на рис. 2.1 с помощью ручных проводок в типовой конфигурации "Бухгалтерский учет".
2. Закройте счет 90, сформировав проводку Д90.9 К99 на сумму торговой наценки.

2.4. Типовая конфигурация "Бухгалтерский учет. Редакция 4.5"

Для ввода информации о совершенных хозяйственных операциях в типовой конфигурации возможно два способа (рис. 2.2). Первый способ — ввести "ручную" операцию — заключается в том, что бухгалтер определяет, какие проводки должны формироваться по первичному (бумажному) документу. Второй способ — ввести электронный аналог первичного документа. При проведении документа программа по заложенному в ней алгоритму сформирует проводки.

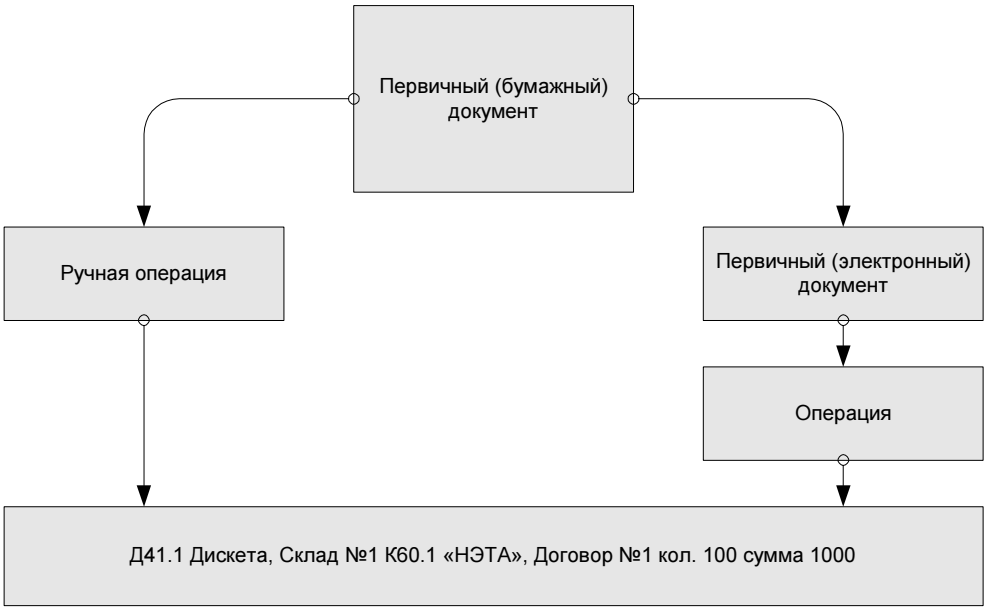


Рис. 2.2. Схема ввода данных в типовой конфигурации "Бухгалтерский учет"

Какой из этих двух способов лучше? Однозначного ответа на этот вопрос нет. Первый способ обладает универсальностью, но требует высокой квалификации бухгалтера и весьма трудоемок, если сумму проводки необходимо рассчитать по некоторому алгоритму. Второй способ фактически требует оператора, который аккуратно заполнит реквизиты электронного документа на основании бумажного документа. Проводки документ сформирует сам. Но здесь существует другая проблема — не для всех первичных документов разработаны электронные аналоги и не все существующие алгоритмы заложены в типовую конфигурацию. Чтобы решить эту проблему, *нужен программист*, который знает встроенный язык системы 1С:Предприятие 7.7 и разбирается в структуре типовой конфигурации настолько, чтобы безболезненно внести требуемые изменения.

Типовая конфигурация предназначена для ведения бухгалтерского учета в "типичной" хозрасчетной организации (для бюджетных организаций есть своя конфигурация). Она позволяет:

- вводить данные о хозяйственных операциях в виде проводок, типовых проводок и документов;
- распечатывать стандартные формы документов (например, приходный кассовый ордер, расходный кассовый ордер, ТОРГ-12 и т. д.);

- получать отчеты о движениях средств в самых разных разрезах: по счетам, по субконто, по датам;
- вести наряду с бухгалтерским налоговый учет в соответствии с налоговым кодексом;
- формировать *регламентные* отчеты, сдаваемые в налоговую инспекцию.

Для реализации рассмотренного примера могут использоваться следующие документы:

- **Поступление товаров** — для ввода приходной накладной от поставщика;
- **Отгрузка товаров, продукции** — для ввода расходной накладной покупателю;
- **Приходный кассовый ордер** — для ввода поступления денег в кассу от покупателя;
- **Расходный кассовый ордер** — для сдачи выручки в банк;
- **Выписка** — для оплаты долга поставщику через расчетный счет;
- **Закрытие месяца** — для закрытия счета 90.

Для отслеживания перемещений денежных средств удобно использовать отчет "Оборотно-сальдовая ведомость". Например, по хозяйственным операциям рассмотренного примера будет получен отчет, приведенный в табл. 2.1.

Таблица 2.1. *Оборотно-сальдовая ведомость за II квартал 2005 г.*

Счет	Наименование	Сальдо на начало периода		Обороты за период		Сальдо на конец периода	
		Дебет	Кредит	Дебет	Кредит	Дебет	Кредит
41	Товары			1,000.00	1,000.00		
50	Касса			1,200.00	1,200.00		
51	Расчетные счета			1,200.00	1,000.00	200.00	
60	Расчеты с поставщиками			1,000.00	1,000.00		
62	Расч. с покупател. и зак.			1,200.00	1,200.00		
90	Продажи			1,200.00	1,200.00		
99	Прибыли и убытки				200.00		200.00
	Итого			**6,800.00**	**6,800.00**	**200.00**	**200.00**

Задание 2.2

1. Отключите проводки у ручных операций (клавиша <F8> в журнале операций).
2. Введите операции, показанные на рис. 2.1, с помощью документов.
3. Сравните проводки, введенные вручную, и проводки, сформированные документами.
4. Постройте отчеты "Оборотно-сальдовая ведомость", "Оборотно-сальдовая ведомость по счету", "Журнал-ордер", "Анализ счета".

2.5. Что не реализовано в типовой конфигурации?

В типовой конфигурации отсутствуют документы учета договоров займа, расчета процентов по договорам займа, отсутствуют документы по учету договоров купли-продажи ценных бумаг. При учете товарно-материальных ценностей реализован только алгоритм расчета себестоимости по средней цене. Эти и другие задачи мы будем решать в следующих главах.

Задание 2.3

Введите следующие операции по учету *договоров займа*. Поступление и возврат денежных средств осуществляется документом "Выписка", а начисление процентов — с помощью ручной операции.

1. Агент Иванов И. И. заключил договор займа с банком Сбербанк РФ (далее *заимодавец*). Согласно договору № ДЗ-70 от 01.04.2005 г. заимодавец должен перечислить на расчетный счет организации сумму 10 000 рублей. Срок договора 6 месяцев и 1 день. При закрытии договора займа в срок начисляется процент годовых 20%, выплачиваемый заимодавцу при возврате денежных средств по окончании срока договора, или 18% годовых при досрочном завершении договора.

Дата	Хозяйственная операция	Бухгалтерские проводки
05.04.2005	Денежные средства от Сбербанка РФ в сумме 10 000 рублей поступили на расчетный счет организации	Д51 К66.3
01.07.2005	Договор завершился. Согласно договору были начислены проценты по ставке 18% годовых	Д91.2 К66.4
01.07.2005	Денежные средства в сумме 10 000 рублей с начисленными процентами были перечислены с расчетного счета организации в банк Сбербанк РФ	Д66.3 К51 Д66.4 К51

2. Агент Иванов И. И. заключил договор займа с банком КМБ (далее заимодавец). Согласно договору № ДЗ-64 от 01.03.2004 г. заимодавец должен перечислить на расчетный счет организации сумму 1000 USD по курсу ЦБ. Срок договора 12 месяцев и 1 день. При закрытии договора займа в срок начисляется процент 10% годовых, выплачиваемый заимодавцу при возврате денежных средств по окончании срока договора, или 8% годовых при досрочном завершении договора

Дата	Хозяйственная операция	Бухгалтерские проводки
03.03.2004	Денежные средства в сумме 1000 USD поступили на расчетный счет организации. Курс ЦБ 28,5321	Д52 К66.33
02.03.2005	Договор завершился. Согласно договору были начислены проценты по ставке 10% годовых. Курс ЦБ 27,7091	Д91.2 К66.44
03.03.2005	Денежные средства в сумме 1000 USD с начисленными процентами были перечислены с расчетного счета организации в банк КМБ. Курс ЦБ 27,6990	Д66.33 К52 Д66.44 К52

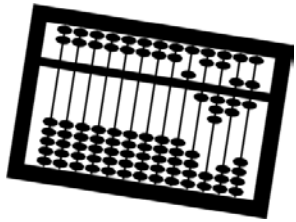
3. Агент Петров П. П. заключил договор займа с банком КМБ (далее заимодавец). Согласно договору № ДЗ-65 от 01.03.2005 г. заимодавец должен перечислить на расчетный счет организации сумму 1000 USD по курсу ЦБ. Срок договора 12 месяцев и 1 день. При закрытии договора займа в срок начисляется процент 12% годовых, выплачиваемый заимодавцу при возврате денежных средств по окончании срока договора, или 9% годовых при досрочном завершении договора

Дата	Хозяйственная операция	Бухгалтерские проводки
01.03.2005	Денежные средства в сумме 1000 USD поступили на расчетный счет организации. Курс ЦБ 27,7007	Д52 К66.33
02.05.2005	Договор завершился. Согласно договору были начислены проценты по ставке 9% годовых. Курс ЦБ 27,7726	Д91.2 К66.44
05.05.2005	Денежные средства в сумме 1000 USD с начисленными процентами по курсу 31,1 были перечислены с расчетного счета организации в банк КМБ. Курс ЦБ 27,7896	Д66.33 К52 Д66.44 К52

2.6. Контрольные вопросы

1. Могут ли корреспондировать в одной проводке балансовый и забалансовый счета?

2. На каком бухгалтерском счете могут, в принципе, учитываться приобретенные дискеты? От чего это зависит?
3. Какие обязательные реквизиты должен иметь первичный документ?
4. Если у организации есть операции в иностранной валюте, то какая ситуация может возникнуть при изменении курса валюты?
5. Какой формулой связаны сальдо начальное, дебетовый оборот, кредитовый оборот и сальдо конечное по бухгалтерскому счету? От чего это зависит?
6. Почему сальдо активного счета не может быть кредитовым?
7. Что такое забалансовый счет? Как можно использовать забалансовые счета?



Глава 3

Изучение встроенного языка программирования

Встроенный язык системы 1С:Предприятие 7.7 предназначен для описания алгоритмов функционирования прикладной задачи. Встроенный язык является предметно-ориентированным. Язык имеет набор предопределенных классов для работы с прикладными объектами: константами, справочниками, документами, операциями и проводками. В конфигурации можно создавать новые виды справочников, новые виды документов, но создать принципиально новый класс объектов невозможно. Этим встроенный язык системы 1С:Предприятие 7.7 отличается от объектно-ориентированных языков.

3.1. Программные модули

Программные модули в конфигурации не являются самостоятельными программами, поскольку являются частью всей конфигурации задачи. Программный модуль — это "контейнер" для размещения текстов процедур и функций, вызываемых системой во время исполнения в определенные моменты времени.

Место размещения конкретного программного модуля предоставляется конфигуратором в тех точках конфигурации задачи, которые требуют описания специфических алгоритмов функционирования.

Каждый отдельный модуль воспринимается системой как единое целое, поэтому все процедуры и функции программного модуля выполняются в одном *контексте*.

Существуют следующие виды программных модулей:

- глобальный модуль;
- модуль формы списка справочника;
- модуль формы группы;
- модуль формы элемента справочника;

- модуль формы документа;
- модуль документа;
- модуль формы журнала документов;
- модуль формы списка счетов;
- модуль формы счета;
- модуль формы журнала операций;
- модуль формы операции;
- модуль формы журнала проводок;
- модуль формы отчета;
- модуль формы обработки;
- модуль расчета.

Особое место занимает глобальный модуль. Он открывается из контекстного меню ветви метаданных **Задача**. Глобальный модуль выполняется при запуске конфигурации (открытии информационной базы). В нем содержатся глобальные переменные, процедуры и функции, доступные из любого другого программного модуля.

Задание 3.1

1. Создайте новую (пустую) информационную базу.
2. Откройте глобальный модуль. Напишите текст программы:
   ```
   Предупреждение("Здравствуй, мир!");
   ```
3. Сохраните конфигурацию, запустите 1С:Предприятие.

3.2. Контекст выполнения программного модуля

Контекст — это множество доступных имен переменных, процедур и функций, доступных в некоторой точке программного модуля. Контекст выполнения программы разделяют на *глобальный* и *локальный*. Глобальный контекст доступен из любого программного модуля и включает:

- значения системных атрибутов, системные процедуры и функции, например, функция `ТекущееВремя()` возвращает текущее системное время, а функция `ТекущаяДата()` — текущую системную дату, установленную на компьютере;
- значения заданных в конфигураторе констант, перечислений, планов счетов и видов субконто, видов и групп расчета, календарей;
- переменные, процедуры и функции глобального модуля с ключевым словом `Экспорт`.

Локальный контекст образуется тем конкретным местом конфигурации задачи, для которого использован этот конкретный модуль. Например, в локальный контекст формы документа входят: контекст формы (атрибут `Форма`, методы формы), реквизиты формы диалога, реквизиты документа, переменные, процедуры и функции, объявленные в программном модуле формы документа.

Задание 3.2

Добавьте к приветствию "Здравствуй, мир!" вывод текущей даты и текущего времени.

3.3. Выполнение программных модулей

Выполнение программных модулей происходит только в режиме 1С:Предприятия. Программный модуль выполняется при открытии формы соответствующего объекта. Глобальный модуль выполняется при запуске системы. Процедуры и функции с предопределенными именами вызываются системой автоматически при наступлении определенного системного события (например, `ПриНачалеРаботыСистемы()`, `ПриЗавершенииРаботыСистемы()` — в глобальном модуле, `ПриОткрытии()`, `ПриЗаписи()`, `ВводНового()` — в модулях форм документов и справочников).

Простейшие программы можно писать в глобальном модуле. Но такая программа выполняется только при загрузке системы, и, в случае изменения текста программы, нужно перезапускать 1С:Предприятие. Поэтому удобнее разрабатывать программные модули во внешних файлах — обработках и отчетах (эти файлы имеют расширение ert и хранятся по умолчанию в каталоге **ExtForms**).

Создать внешний отчет (обработку) можно в конфигураторе. Заходим в меню **Файл** и выбираем пункт **Новый**. Выбираем вид отчета **Внешний отчет (обработка)**. Открывается форма отчета (обработки), состоящая из четырех вкладок:

- **Диалог**;
- **Модуль**;
- **Описание**;
- **Таблица**.

В диалоге формы мы можем размещать различные интерфейсные элементы: кнопки, текст, поля для ввода, списки, таблицы значений, рисунки. Для вставки элемента диалога нужно выбрать пункт из меню **Вставить** или аналогичную кнопку на панели инструментов. Автоматизировать некоторые стандартные действия при добавлении элемента диалога можно, выбрав

пункт **Элемент Диалога**. Например, при добавлении кнопки можно сразу же создать процедуру, которая будет вызываться при нажатии этой кнопки. Сопоставление процедуры и элемента диалога производится следующим образом. У элемента диалога в свойствах обычно есть вкладка **Дополнительно**, на которой можно написать формулу на встроенном языке. Эта формула будет выполняться при работе с этим элементом, например, при нажатии кнопки, при редактировании поля для ввода.

В модуле формы мы пишем программу. Действия, выполняемые отчетом (обработкой), лучше всего оформлять в виде процедур и вызывать их нажатием какой-нибудь кнопки.

У элемента диалога есть **Идентификатор**, по которому мы обращаемся к значению (значениям) данного элемента. Работа с идентификатором аналогична работе с переменной. С помощью идентификатора элемента диалога можно управлять его свойствами через системную переменную `Форма`.

Выполнение программного модуля возможно только в режиме 1С:Предприятие. Открыть объект можно несколькими способами:

- через меню **Операции** (внешние отчеты через меню **Файл**, пункт **Открыть**);
- через меню и кнопки пользовательского интерфейса;
- программно: системной функцией `ОткрытьФорму(<ОписательОбъекта>,<Параметр>)`. В первом параметре мы указываем либо строку — описание объекта (например, `"Отчет.ОкноВМир"`), либо ссылку на существующий объект (элемент справочника, документ и т. п.). Во втором параметре мы можем передать в форму переменную любого типа, например, список значений. Эта переменная будет доступна в открытой форме в виде реквизита формы: `Форма.Параметр`.

Задание 3.3

Создайте внешний отчет "ОкноВМир", который по нажатию кнопки **Сформировать** выводит текст "Здравствуй, мир!".

3.4. Формат операторов

Текст программного модуля состоит из операторов и комментариев. Комментарии начинаются с символов //. Операторы имеют следующий формат:

```
[~метка:]Оператор [(Параметры)][ДобавочноеКлючевоеСлово];
```

В начале может находиться метка, на которую можно передать управление оператором `Перейти`. У оператора может быть задан набор параметров, а также добавочное ключевое слово: `Далее` или `Экспорт`. Каждый оператор может располагаться на нескольких строках, но обязательно должен заканчиваться точкой с запятой (`;`).

3.5. Имена переменных, процедур и функций

Именем переменной, процедуры или функции может быть любая последовательность букв, цифр и знаков подчеркивания, начинающаяся с буквы или со знака подчеркивания. Регистр букв не учитывается. Допускается использовать как русские, так и английские буквы, что иногда приводит к "необъяснимым" ошибкам, когда кажется, что оператор написан правильно, а система выдает ошибку. Так как есть одинаковые по написанию буквы (например, "с", "о", "а" и др.), то переменные "a1", где а — русская буква, и "a1", где а — английская буква, будут восприниматься системой как разные.

Внимание!

Вновь создаваемые имена не должны совпадать с уже существующими, доступными на момент выполнения модуля. *Чтобы определить, является ли слово зарезервированным, нужно воспользоваться поиском в синтакс-помощнике:* установить курсор на проверяемое слово и одновременно нажать комбинацию клавиш <Ctrl>+<F1>. Таким же способом можно узнать синтаксис системной процедуры или функции: количество и назначение параметров, значения по умолчанию, возвращаемое значение.

3.6. Структура программного модуля

Программный модуль имеет следующую структуру:

- раздел определения переменных;
- раздел процедур и функций;
- раздел основной программы.

Некоторые разделы могут отсутствовать, но важно соблюдать последовательность этих разделов (листинг 3.1).

Листинг 3.1. Разделы программного модуля

```
// Определение переменных
Перем ЭтоПерваяПеременная;
Перем ЭтоВтораяПеременная;
// Процедуры и функции
Процедура ЭтоПроцедура()
  // текст процедуры
КонецПроцедуры
Функция ЭтоФункция()
   // текст функции
```

```
КонецФункции
// Раздел основной программы
ЭтоПерваяПеременная = "123";
```

Внимание!

После слов `КонецПроцедуры` и `КонецФункции` точка с запятой не ставится!

3.7. Процедуры и функции программного модуля

Процедуры и функции можно классифицировать следующим образом.

- *Системные* — функции и процедуры, предоставляемые системой 1С:Предприятие 7.7. Например, функция `Сообщить(<текст_сообщения>)` выводит текст в окно сообщений. Полный перечень системных функций можно посмотреть в синтакс-помощнике в разделе **Встроенный язык/Системные/Функции и процедуры**;
- *Предопределенные* — функции и процедуры, имена которых предопределены системой в том смысле, что функция или процедура с таким именем будет вызвана системой при наступлении определенного события. Например, процедура `ПриНачалеРаботыСистемы()`, определенная в глобальном модуле, будет вызвана при запуске 1С:Предприятия. Процедура `ОбработкаПроведения()`, определенная в модуле документа, будет вызвана при проведении документа;
- *Пользовательские* — процедуры и функции, определяемые пользователем.

3.7.1. Описание процедуры

Описание процедуры имеет следующий формат:

```
Процедура <Имя_процедуры>([Знач]<Параметр1>[[=<ЗначениеПоУмолчанию1>],
…])[Экспорт][Далее]
// Объявления локальных переменных;
// Операторы;
[Возврат;]
// Операторы;
КонецПроцедуры
```

Ключевое слово `Знач` означает, что фактические параметры передаются в процедуру по значению. Если ключевое слово `Знач` отсутствует, то фактические параметры передаются в процедуру по ссылке, и изменение значений параметров в процедуре сохранится при возврате из процедуры. Если задано

значение по умолчанию, то данный параметр при вызове можно опускать, тогда формальный параметр примет указанное по умолчанию значение.

Ключевое слово `Экспорт` имеет смысл только в глобальном модуле и означает, что процедура будет доступна в любом программном модуле.

Ключевое слово `Далее` означает, что это описание только заголовка процедуры (после слова `Далее` точка с запятой не ставится!), а тело процедуры будет описано далее. Необходимость описания заголовка процедуры вызвана тем, что интерпретатор встроенного языка однопроходный, и если вызов процедуры в тексте программного модуля идет выше ее описания, то будет выдана ошибка.

В листинге 3.2 приведен пример процедуры, вычисляющей сумму как произведение цены на количество.

Листинг 3.2. Пример процедуры

```
Процедура ВычислитьСумму(Цена, Количество, Сумма)
        Сумма=Цена*Количество;
КонецПроцедуры
```

3.7.2. Описание функции

Описание функции имеет такую же структуру, как и процедура, но, в отличие от процедуры, функция должна возвращать какое-нибудь значение.

```
Функция <Имя_функции>([Знач]<Параметр1>[[=<ЗначениеПоУмолчанию1>], …])
[Экспорт][Далее]
// Объявления локальных переменных;
// Операторы;
Возврат <Значение>;
КонецФункции
```

Приведем пример функции в листинге 3.3.

Листинг 3.3. Пример функции

```
Функция Сумма(Цена=10, Количество=100)
        Возврат Цена*Количество;
КонецПроцедуры
```

3.7.3. Вызов процедур и функций

Для того чтобы вызвать процедуру или функцию, нужно указать имя процедуры или функции и в скобках перечислить список фактических параметров.

Даже если у процедуры или функции нет параметров, или все параметры используются по умолчанию, нужно поставить круглые скобки, так как именно по ним система отличает вызов процедуры или функции от переменной.

В листинге 3.4 приведены различные варианты вызова процедуры `ВычислитьСумму()` и функции `Сумма()`. Все перечисленные способы вычисления суммы дадут один и тот же результат.

Листинг 3.4. Различные варианты вызова функций и процедур

```
Перем Сум;
ВычислитьСумму(10, 100, Сум); // Вызов процедуры
Сум=Сумма(10, 100); // Вызов функции
Сум=Сумма(,100); // Вызов функции, опущен первый параметр
Сум=Сумма(10); // Вызов функции, опущен второй параметр
Сум=Сумма(); // Вызов функции, опущены оба параметра
```

Задание 3.4

Модифицируйте глобальный модуль таким образом, чтобы при выходе из программы сообщалось, сколько времени проработал пользователь.

Указание

Воспользуйтесь функциями `ТекущаяДата()` и `ТекущееВремя()`. Момент входа в систему нужно запоминать в предопределенной процедуре `ПриНачалеРаботыСистемы()`, а продолжительность работы надо вычислять в предопределенной процедуре `ПриЗавершенииРаботыСистемы()`.

3.7.4. Рекурсивный вызов

В 1С:Предприятии 7.7 допускается рекурсивный вызов функции самой себя. В листинге 3.5 приведен пример вычисления факториала с использованием рекурсии.

Листинг 3.5. Вычисление функции Н!=1*2*...*Н

```
Функция Факториал(Н)
        Если Н=0 Тогда
                Возврат 1;
        КонецЕсли;
        Возврат Н*Факториал(Н-1);
КонецФункции
```

Задание 3.5

Создайте внешний отчет "Факториал", который рассчитывает значение факториала для заданного числа (задать его как реквизит диалога формы) и выводит в окно сообщений.

3.7.5. Передача локального контекста в процедуру

Для передачи локального контекста используется ключевое слово `Контекст` в качестве параметра. Допустим, у нас есть несколько различных документов, причем у всех есть реквизиты "Цена", "Количество", "Сумма". В каждом программном модуле формы редактирования документа нужна процедура, вычисляющая сумму по введенной цене и количеству. В глобальном модуле напишем процедуру, приведенную в листинге 3.6.

Листинг 3.6. Пример передачи в качестве параметра процедуры локального контекста

```
Процедура глВычислитьСумму(Конт) Экспорт
   Конт.Сумма = Конт.Цена*Конт.Количество;
КонецПроцедуры
```

Здесь переменная `Конт` является формальным параметром, которому при вызове процедуры будет присвоен локальный контекст. В данном примере обращение к реквизитам и методам локального контекста происходит "через точку" после идентификатора `Конт`.

Теперь в любом программном модуле конфигурации (в данном примере, в любом модуле формы документа) для вычисления суммы можно вызвать процедуру:

```
глВычислитьСумму(Контекст);
```

Замечание

Префикс "`гл`" в названии глобальных процедур показывает, что искать описание этой процедуры нужно в глобальном модуле. Это соглашение используется разработчиками типовых конфигураций и рекомендуется для использования другим программистам.

3.8. Типы данных

В системе поддерживаются базовые и агрегатные типы данных.

3.8.1. Базовые типы данных

К базовым типам данных относятся число, строка и дата.

С числовыми переменными допустимо проведение арифметических операций +, -, *, /, а также действие математических функций Окр() — округлить, Цел() — взять целую часть, Лог() — вычислить натуральный логарифм.

Замечание

Чтобы вычислить квадратный корень, экспоненту, синус или косинус, придется либо раскладывать данные функции в степенной ряд, либо воспользоваться вызовом внешней компоненты. Первый способ подробно описан на диске ИТС 1С:Предприятие. Работаем с программами в разделе "Методическая поддержка 1С:Предприятия 7.7", статья "Реализация математических функций на встроенном языке 1С:Предприятия".

Для строковых величин допустима операция конкатенации (сложения двух строк). Например, выражение "абв" + "где" даст в результате "абвгде".

Для переменных типа "Дата" допустима операция вычитания, которая дает в результате число дней между двумя датами. Например, выражение '01.01.2005'-'01.01.2004' даст в результате число 366 (все правильно, 2004 год был високосным!).

Внимание!

Во встроенном языке программирования системы 1С:Предприятие строковая константа задается в двойных кавычках, а константа типа "Дата" — в одинарных.

Если к переменной типа "Дата" добавить число, то мы получим дату, отстоящую от начальной на заданное число дней. Например, выражение '01.01.2004'+366 даст в результате дату '01.01.2005'. Чтобы получить дату, отстоящую от начальной на заданное число месяцев, удобно использовать системную функцию ДобавитьМесяц(). Например, выражение ДобавитьМесяц('01.01.2004',12) даст в результате дату '01.01.2005'.

Подсказка

Чтобы интерактивно вычислить какое-нибудь выражение, можно вызвать **Табло** из меню **Сервис** в режиме "1С:Предприятие" или из меню **Отладка** в режиме "Отладчик".

Для преобразования значения одного базового типа в другой используются функции

- Строка(<Значение>);
- Число(<Значение>);
- Дата(<Значение>);
- Дата(<Год>,<Месяц>,<День>).

Если в выражении используются величины разных типов, то выражение приводится к типу первого операнда.

Типизация переменных в языке не жесткая, т. е. тип переменной определяется ее значением. В листинге 3.7 показано, как тип переменной `Пер1` последовательно принимает значения `"Неопределенный"`, `"Число"`, `"Строка"`, `"Дата"`.

Листинг 3.7. Пример изменения типа переменной

```
Перем Пер1; // Объявление переменной
Пер1 = 123; // Тип переменной — число
Пер1 = "123"; // Тип переменной — строка
Пер1 = '01.01.2005'; // Тип переменной — дата
```

3.8.2. Агрегатные типы данных

Агрегатные типы данных — это специализированные типы данных, предназначенные для работы с объектами 1С:Предприятия. Перечислим некоторые из них:

- `Константа` — средство работы с условно постоянными значениями. В константах хранится информация, которая не изменяется вообще или изменяется достаточно редко;
- `Справочник` — средство для ведения списков однородных элементов данных;
- `Перечисление` — средство работы с элементами данных, список возможных значений которых жестко задан на этапе разработки конфигурации;
- `Документ` — средство для ввода первичной информации о совершаемых хозяйственных операциях;
- `Запрос` — средство для выполнения обращения к документам, регистрам, документам, справочникам и журналам расчетов с целью получения сводной информации при формировании выходных отчетов;
- `Текст` — средство работы с текстовыми документами. Таблица — средство работы с таблицами (отчетами);
- `СписокЗначений` — средство для создания списка значений каких-либо данных и возможность в дальнейшем сортировать и выбирать нужные значения из списка;
- `ТаблицаЗначений` — средство для создания списка значений каких-либо данных и возможность в дальнейшем сортировать и выбирать нужные значения из списка;

- `Картинка` — средство для работы с графическими файлами;
- `Периодический` — средство для работы с периодическими реквизитами справочников и периодическими константами;
- `ФС` — средство для работы с дисковыми файлами непосредственно из встроенного языка системы 1С:Предприятие;
- `XBase` — средство для работы с файлами баз данных DBF-формата непосредственно из встроенного языка системы 1С:Предприятие.

Переменную агрегатного типа можно создать функцией `СоздатьОбъект("Тип")`, у которой в качестве параметра указать строку — название типа.

Каждый агрегатный тип данных имеет свой набор атрибутов и методов. Атрибуты по свойствам напоминают переменные, то есть им можно присваивать, или читать, их значения. Методы — это те действия, которые может выполнять агрегатный тип данных. Получить доступ к атрибутам и методам переменной агрегатного типа можно через точку, например, `ПеременнаяАгрегатногоТипа.Атрибут` или `ПеременнаяАгрегатногоТипа.Метод()`.

Подсказка

Запоминать все атрибуты и методы агрегатных объектов не нужно — их всегда можно посмотреть через синтакс-помощник, который вызывается через меню **Сервис** в конфигураторе.

Работу с агрегатными типами данных мы рассмотрим на примере списка значений и таблицы значений в *разд. 3.10, 3.11 данной главы*.

3.9. Управляющие операторы

Для управления логикой выполнения программы используются логические выражения и *управляющие операторы*. Логическое выражение записывается с помощью символов сравнения = (равно), <> (не равно), > (больше), < (меньше), <= (меньше либо равно), >=(больше или равно), при этом сравнение производится только над значениями одинаковых типов (то есть нельзя, например, сравнивать число и строку).

Из нескольких сравнений можно построить более сложное условие с помощью булевых операций И, ИЛИ, НЕ. При этом простые логические выражения должны быть заключены в круглые скобки. Например:

- истина: `1=1`;
- ложь: `1<>1`;
- сложное условие: `(1=1) или (1<>1)`.

Внимание!

Логическое выражение всегда выполняется до конца, даже если результат очевиден заранее. В рассмотренном примере выражение в первых скобках истинно И, так как первое и второе выражение связывает ИЛИ, то результат вычисления выражения во вторых скобках не влияет на результат вычисления всего выражения. Однако система все равно вычислит логическое выражение во вторых скобках. Конечно, на производительность это практически не влияет, но может привести к ошибке, если вычисление второго условия возможно только при истинности первого условия. Например, в первом условии проверяется существование объекта, а во втором используется атрибут или метод этого объекта. Если объект не существует, то попытка получить атрибут этого объекта или выполнить метод приведет к ошибке типа "Поле агрегатного объекта не обнаружено".

3.9.1. Оператор ветвления

Оператор ветвления имеет следующий формат:

```
Если <Логическое_выражение1> Тогда
        // Операторы 1
[ИначеЕсли <Логическое_выражение2> Тогда]
        // Операторы 2
[ИначеЕсли <Логическое_выражение3> Тогда]
        // Операторы 3
…
[Иначе]
        // Операторы
КонецЕсли;
```

3.9.2. Циклы

В 1С:Предприятии 7.7 есть два вида цикла: `Пока` и `Для`

Цикл *Пока*

```
Пока <Логическое_выражение> Цикл
…
[Прервать]
…
[Продолжить]
…
КонецЦикла;
```

Цикл *Для*

```
Для <Имя_переменной> = <Выражение1> По <Выражение2> Цикл
…
[Прервать]
…
[Продолжить]
…
КонецЦикла;
```

Величина приращения счетчика при каждом выполнении цикла равна 1. Условие выполнения цикла всегда проверяется вначале, перед выполнением цикла. С помощью оператора `Прервать` можно прекратить выполнение цикла и передать управление на оператор, следующий после слова `КонецЦикла`. С помощью оператора `Продолжить` можно прервать выполнение текущей итерации цикла и перейти на следующую итерацию. Приведем пример вычисления факториала для целых чисел в интервале от 1 до 10 (листинг 3.8).

Листинг 3.8. Вычисление факториала от 1 до 10

```
Для Н=1 По 10 Цикл
   Сообщить(""+Н+"! = "+Факториал(Н));
КонецЦикла;
```

Задание 3.6

В отчет "Факториал" добавьте кнопку, которая выводит в окно сообщений значение факториала в интервале от `Н1` до `Н2` (задать их, как реквизиты диалога формы). Какое максимальное значение может принимать число `Н2`?

3.9.3. Обработка ошибок

Бывают ситуации, когда правильность выполнения оператора зависит от контекста в момент выполнения. Например, выражение `Сумма/Количество` будет вычисляться правильно, если переменная `Количество` не равна нулю. Если при выполнении программного модуля возникает ошибка, то выполнение модуля прерывается, и управление передается в главное меню системы 1С:Предприятие 7.7. Чтобы корректно обрабатывать данную ситуацию, существует специальная конструкция `Попытка`, имеющая следующий формат

```
Попытка
…
Исключение
…
КонецПопытки;
```

При возникновении ошибки управление передается на операторы после слова `Исключение`. Если ошибки не произошло, то управление передается на оператор, следующий после выражения `КонецПопытки`. Рассмотрим пример (листинг 3.9). В программе вычисляется цена путем деления суммы на количество. Если количество вдруг окажется нулевым, то при выполнении возникнет ошибка `Деление на ноль`, и выполнение программы прервется. Чтобы этого не произошло, поместим оператор вычисления суммы в конструкцию `Попытка` и, в случае ошибки, зададим соответствующее сообщение.

Листинг 3.9. Применение конструкции `Попытка`

```
Попытка
        Цена = Сумма/Количество;
Исключение
        Сообщить("Количество равно нулю!")
        Возврат;
КонецПопытки;
```

3.10. Работа с объектом *"СписокЗначений"*

Объект `"СписокЗначений"` применяется для создания динамических списков (не сохраняемых в базе данных), которые могут отображаться в диалоговых формах (элементы "Список" и "Поле со списком") для выбора одного или нескольких значений из списка. Элемент списка содержит три поля: значение, представление и пометка. Значение может принимать значение любого типа, представление же всегда имеет тип "Строка". В форме диалога отображается представление (если оно задано). Пометка означает, помечено данное значение или нет.

Для работы со списком значений существует агрегатный тип данных `СписокЗначений`. Переменная типа `"СписокЗначений"` создается с помощью функции `СоздатьОбъект("СписокЗначений")`, либо визуально при добавлении на форму диалога элементов "Список" или "Поле со списком".

Первоначально список значений пуст. Для добавления нового элемента в список используется функция `ДобавитьЗначение(<Значение>,<Представление>)`. В листинге 3.10 приведен пример создания и заполнения списка.

Листинг 3.10. Пример создания и заполнения списка значений

```
Сп=СоздатьОбъект("СписокЗначений");
Сп.ДобавитьЗначение("Иванов", "Фамилия"); // 1-е значение
```

```
Сп.ДобавитьЗначение("Иван", "Имя"); // 2-е значение
Сп.ДобавитьЗначение("Иванович", "Отчество"); // 3-е значение
Сп.ДобавитьЗначение(1000, "Оклад"); // 4-е значение
Сп.ДобавитьЗначение('01.01.2005', "ДатаПриема"); // 5-е значение
```

Чтобы извлечь значение из списка по его порядковому номеру, существует функция `ПолучитьЗначение(<Номер>)`. Элементы списка нумеруются с единицы. Количество элементов в списке можно узнать с помощью функции `РазмерСписка()`. В листинге 3.11 приведен пример, выводящий все элементы списка в окно сообщений.

Листинг 3.11. Пример, выводящий все элементы списка в окно сообщений

```
Представление= "";
Для Н=1 По Сп.РазмерСписка() Цикл
   Значение=Сп.ПолучитьЗначение(Н, Представление);
   Сообщить(Представление+ " "+Значение);
КонецЦикла;
```

Поскольку в списке могут находиться значения произвольного типа, то в качестве значения элемента списка может быть добавлен другой список значений, например, `Сп.ДобавитьЗначение(СоздатьОбъект("СписокЗначений"))`. Таким образом, на основании списка значений можно построить древовидную структуру.

Список значений имеет достаточно большое количество сервисных функций, с которыми вы можете ознакомиться через синтакс-помощник. Среди них такие как сортировка, проверка принадлежности списку, удаление всех элементов из списка, установка и чтение пометки и др.

С помощью списка можно организовать динамическое меню. Для этого нужно заполнить список пунктами меню и открыть его функцией `ВыбратьЗначение()`.

3.11. Работа с объектом *"ТаблицаЗначений"*

Объект `"ТаблицаЗначений"` применяется для создания динамических массивов (не сохраняемых в базе данных), которые могут отображаться в диалоговых формах (элемент "Таблица значений"). Таблица значений создается с помощью функции `СоздатьОбъект("ТаблицаЗначений")` либо визуально при добавлении в форму диалога элемента "Таблица значений". В листинге 3.12 приведен пример создания и заполнения таблицы с колонками `"Товар"`, `"Цена"`, `"Количество"`, `"Сумма"`.

Листинг 3.12. Пример создания и заполнения таблицы с колонками

```
Т=СоздатьОбъект("ТаблицаЗначений")
// Определяем идентификаторы и параметры колонок
Т.НоваяКолонка("Товар","Строка",30,,"Наименование товара",20);
Т.НоваяКолонка("Цена","Число",10,2,"Цена",10);
Т.НоваяКолонка("Количество","Число",10,3,"Количество",10);
Т.НоваяКолонка("Сумма","Число",15,3,"Сумма",10);
// Создаем новую строку
Т.НоваяСтрока();
Т.Товар = "Дискета";
Т.Цена = 10;
Т.Количество = 100;
Т.Сумма = Т.Цена * Т.Количество;
```

Перебрать все строки таблицы значений можно следующим образом (листинг 3.13).

Листинг 3.13. Перебор строк таблицы значений

```
Т.ВыбратьСтроки();
Пока Т.ПолучитьСтроку()=1 Цикл
   Сообщить(Т.Товар + " количество " + Т.Количество);
КонецЦикла;
```

Этот прием обхода будет постоянно применяться для перебора различных коллекций значений в дальнейшем.

Для таблицы значений есть много различных сервисных функций. Отсортировать строки таблицы можно функцией `Сортировать("Колонки")`, у которой в строковом параметре перечисляются колонки, разделенные запятыми, по которым производится сортировка. Если к имени колонки добавить символ "+", то сортировка выполняется по возрастанию значений, если "-" — то по убыванию.

Функция `Свернуть("ГруппКолонки", "СуммКолонки")` выполняет свертку строк таблицы значений таким образом, что заменяет на одну все дублирующие (по значениям группировочных колонок) строки, суммируя значения по суммируемым колонкам. В листинге 3.14 приведен пример свертки и сортировки таблицы.

Листинг 3.14. Группировка и свертка таблицы значений

```
Т.Свернуть("Товар,Цена", "Количество");
Т.Сортировать("Товар+,Цена-");
```

Задание 3.7

Создайте обработку "Работа с таблицей значений".

1. В форму диалога поместите таблицу значений.
2. В модуле формы опишите колонки таблицы: **Товар**, **Количество**, **Цена**, **Сумма**.
3. Поместите на форму диалога реквизиты диалога "Товар", "Цена", "Количество" и кнопку **Добавить строку**, которая добавляет новую строку в таблицу значений и заполняет ее значениями соответствующих реквизитов. Сумма должна вычисляться как произведение цены на количество.
4. Добавьте кнопку **Удалить строку**, которая удаляет текущую строку таблицы значений. Предусмотрите ситуацию пустой таблицы значений.
5. Добавьте кнопку **Свернуть**, которая должна суммировать по товарам сумму и количество, в колонке **Цена** должна вычисляться средняя цена товара.
6. Добавьте кнопку **Развернуть**, которая восстанавливает состояние таблицы значений до свертки.
7. Добавьте кнопку **Сортировка** для упорядочивания строк по товарам.

3.12. Запуск внешних приложений из 1С:Предприятия

Для запуска внешнего приложения можно использовать функцию `ЗапуститьПриложение(<Строка команды>)`. Выполнение этого оператора эквивалентно тому, что вы вручную откроете в Windows меню **Пуск**, пункт **Выполнить** и наберете строку команды.

Для запуска и управления внешним приложением можно использовать механизм OLE Automation. В листингах 3.15—3.17 приведены примеры использования этого механизма при работе с программами Microsoft Word и Microsoft Excel.

Листинг 3.15. Запись информации в таблицу Microsoft Excel

```
Окно = СоздатьОбъект("Excel.Application");
Окно.Visible = 1; // Делаем окно видимым
Окно.Caption = "Отчет"; // Задаем имя окну
```

```
Окно.Workbooks.Add(); // Создаем новую рабочую книгу
Для Н=1 По 10 Цикл
Ячейка = Окно.Cells(Н,1);
Ячейка.Value = Н;
КонецЦикла;
```

Листинг 3.16. Чтение информации из таблицы Microsoft Excel

```
ОкноЭксел=СоздатьОбъект("Excel.Application");
ОкноЭксел.Workbooks.Open("C:\Книга\Сотрудники.xls");
Док=ОкноЭксел.Worksheets(1);
// Определяем число рядов
КоличествоРядов=ОкноЭксел.ActiveSheet.UsedRange.Rows.Count;
// Определяем число колонок
КоличествоКолонок=ОкноЭксел.ActiveSheet.UsedRange.Columns.Count;
Для Ряд=1 По КоличествоРядов Цикл
        Для Кол=1 По КоличествоКолонок Цикл
                Значение=Док.Cells(Ряд, Кол).Value;
                Сообщить("Значение ("+Ряд+","+Кол+") = "+Значение);
        КонецЦикла;
КонецЦикла;
```

Листинг 3.17. Чтение информации из документа Microsoft Word

```
ОкноВорд=СоздатьОбъект("Word.Application"); // Создаем объект
ОкноВорд.Documents.Open ("C:\Книга\Глава_1.doc"); // Открываем файл
Док=ОкноВорд.ActiveDocument; // Получаем объект документа
КоличествоСтрок=Док.Paragraphs.Count(); // Количество параграфов (строк).
Для Н=1 По КоличествоСтрок Цикл
   Параграф=Док.Paragraphs(Н).Range.Text; // Получаем строку
   Сообщить(Параграф);
КонецЦикла;
ОкноВорд.Quit(); // Выход
```

Задание 3.8

Создайте внешний отчет "Работа с офисными документами".

1. Добавьте в форму диалога кнопку **Открыть Excel**. При нажатии этой кнопки вызывается Excel. Используя конструкцию `Попытка`, программа

будет выдавать сообщение "Excel недоступен" при неудачной попытке открытия внешней программы.

2. Добавьте в форму диалога кнопку **Открыть Word**. При нажатии этой кнопки вызывается Word.
3. Можно воспользоваться рассмотренными примерами для чтения данных из файлов Word и Excel.

3.13. Работа с транзакциями

Транзакции применяются для выполнения длительных и критических для функционирования системы операций. В некоторых случаях система сама начинает транзакцию, и специально описывать ее не надо. Например, это происходит при проведении документов: так как в программе могут работать одновременно несколько пользователей, то при записи документа (например, при продаже товара со склада) недопустима ситуация "одновременного" проведения двух документов — один из них должен быть проведен раньше, чем другой (иначе можно продать один и тот же товар дважды!). Поэтому система выполняет проведение документа в режиме транзакции, при котором база данных становится недоступной для записи остальным пользователям. Естественно, что при большом количестве пользователей, которые активно вводят данные в программу, работа системы замедляется. Одним из решений этой проблемы может быть установка версии для SQL, в этом случае чтение и запись в базу данных выполняются средствами MS SQL Server.

В остальных случаях можно транзакцию начать и зафиксировать с помощью следующих функций:

```
НачатьТранзакцию();
...
[ОтменитьТранзакцию();]
...
ЗафиксироватьТранзакцию();
```

Отмена транзакции применяется в случае обнаружения некорректной ситуации. При этом все изменения, внесенные в базы данных с момента начала транзакции, отменяются. Чтобы зафиксировать изменения, необходимо выполнить функцию `ЗафиксироватьТранзакцию()`.

Задание 3.9

1. Вставьте разработанные внешние отчеты в конфигурацию.
2. Добавьте кнопки на панель инструментов для вызова разработанных отчетов.

Задание 3.10

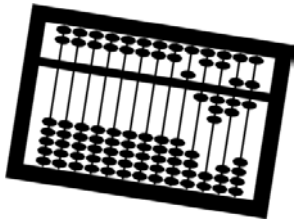
1. Напишите процедуру для вычисления экспоненты числа *x*. Для вычисления воспользуйтесь приближенной формулой (где *n* достаточно велико).

$$e^x = 1 + x + 1/2 * x^2 + ... + 1/n! \quad x^n.$$

2. Напишите процедуру для вычисления степенной функции x^y, используя процедуру для вычисления экспоненты.

3.14. Контрольные вопросы

1. Можно ли с использованием системы 1С:Предприятие 7.7 написать программу, работающую независимо от системы?
2. Чем глобальный модуль отличается от остальных программных модулей?
3. Чем отличается локальный контекст от глобального?
4. Как запустить программный модуль на выполнение?
5. Почему имя переменной не может начинаться с цифры?
6. Что произойдет, если после слова `КонецПроцедуры` поставить точку с запятой?
7. Зачем используется ключевое слово `Далее`?
8. Для чего используются предопределенные процедуры?
9. Чем отличается передача параметров в процедуру по ссылке и по значению?
10. Можно ли в типовой конфигурации определить по названию процедуры, что она описана в глобальном модуле?
11. Что такое транзакция и когда возникает необходимость ее применения?



Глава 4

Работа с константами, справочниками и документами

Все объекты системы 1С:Предприятие 7.7, в которых хранятся данные, можно разбить на базовые и прикладные. *Базовые* объекты присутствуют всегда, а *прикладные* только при наличии определенной компоненты. В этой главе мы рассмотрим базовые объекты: константы, справочники и документы.

4.1. Объекты системы 1С:Предприятие 7.7

Задача любой программы — ввод исходных данных, обработка данных по заданному алгоритму и вывод результатов. Спецификой любой учетной системы является то, что данные вводятся непрерывно в течение длительного времени, часто ввод осуществляется в момент совершения хозяйственной операции. Поэтому одна из функций системы 1С:Предприятие — хранение информации. Ввод и хранение информации может быть, в зависимости способа ввода и места хранения, следующих видов:

- в режиме "Конфигуратор" программистом вводятся виды констант, справочников, документов, а также перечисления, планы счетов, бухгалтерские счета, виды регистров и расчетов, группы и журналы расчетов, календари. Место хранения этой информации — конфигурация;
- в режиме "1С:Предприятие" пользователем вручную вводятся значения констант, элементы справочников, документы, операции и проводки, бухгалтерские счета, значения календарей. Место хранения этой информации — таблицы базы данных;
- в режиме "1С:Предприятие" при проведении документов программно создаются операции и проводки, движения регистров, записи журналов расчетов. Место хранения этой информации — таблицы базы данных.

Как видите, некоторые объекты могут вводиться только в конфигураторе (например, перечисления), некоторые — только в 1С:Предприятии (справочники, документы), некоторые — разными способами (бухгалтерские счета).

Система 1С:Предприятие 7.7 предоставляет возможность хранить информацию в базе данных в виде констант, справочников, документов, планов счетов, операций с проводками, записей журнала расчетов, календарей. Отметим, что перечисления, элементы справочников, бухгалтерские счета, документы, виды расчетов, календари являются объектами, на которые можно ссылаться.

Например, в качестве типа реквизита может выступать тип "Справочник", хотя корректнее сказать (и это сделано в восьмой версии 1С:Предприятия), *ссылка* на элемент справочника. Это означает, что каждый элемент справочника имеет уникальный внутренний идентификатор (ссылку), по которому можно найти в базе данных соответствующие ему записи одной или нескольких таблиц данных. Если удалить элемент справочника и создать новый элемент с такими же значениями реквизитов, то новый элемент будет отличаться от удаленного своим внутренним идентификатором.

Отсюда следует, что при удалении данных (а также при различных программных и аппаратных сбоях, связанных с потерей данных) может возникнуть проблема ссылочной целостности: есть ссылка, но нет объекта с таким идентификатором. При тестировании и исправлении информационной базы *(см. главу 1)* есть два варианта решения этой проблемы:

- очистка ссылок на несуществующие объекты;
- создание "пустых" объектов по этим ссылкам.

Результат такого восстановления вы можете обнаружить, если вдруг появились элементы с названиями типа ФС-1, ФС-2 и т. д.

4.2. Константы

Константы — это средство для хранения условно постоянной информации, задаваемой пользователем. Чтобы добавить новую константу, необходимо открыть конфигурацию, в разделе метаданных **Константы** нажать правую кнопку мыши, выбрать пункт **Новая Константа**. Далее задать идентификатор константы и тип значения, как показано на рис. 4.1.

Задать значение константы можно только в режиме "1С:Предприятие". Для этого нужно открыть меню **Операции** и выбрать пункт **Константы**. А как же быть, если константе нужно присвоить какое-то начальное значение до того, как пользователь приступит к работе с программой?

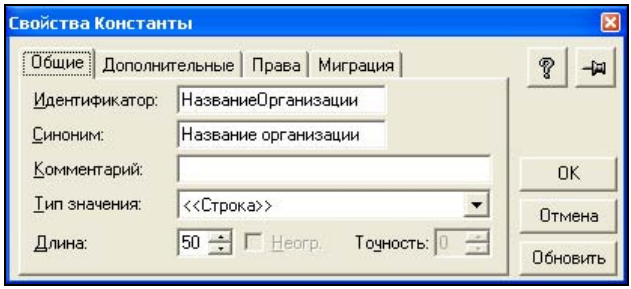


Рис. 4.1. Свойства константы

Для этого можно предложить следующий способ. Добавим константу `"НомерРелиза"`. По умолчанию она будет иметь пустое значение. В предопределенной процедуре `ПриНачалеРаботыСистемы()` в глобальном модуле вставим текст, приведенный в листинге 4.1.

Листинг 4.1. Задание начальных значений констант

```
Процедура ПриНачалеРаботыСистемы()
Если ПустоеЗначение(Константа.НомерРелиза)=1 Тогда
        Константа.НазваниеОрганизации =  "ООО СтройТоргВсе";
        // Аналогично задаем остальные константы
        Константа.НомерРелиза = 1;
КонецЕсли;
КонецПроцедуры
```

Системная функция `ПустоеЗначение(<Значение>)` сравнивает параметр `<Значение>` со значением, которым по умолчанию инициализируется переменная данного типа.

Отметим еще одно важное свойство констант — они могут быть периодическими. Для этого в свойствах константы на вкладке **Дополнительные** нужно установить флажок **Периодический** (рис. 4.2).

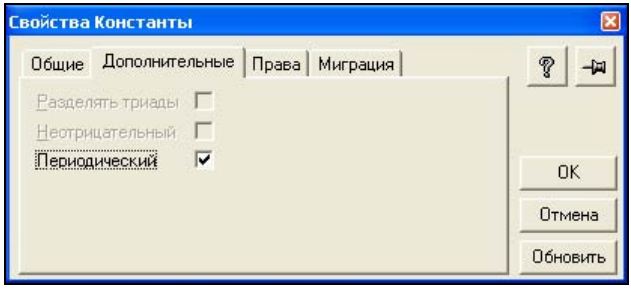


Рис. 4.2. Дополнительные свойства константы

При работе с периодической константой необходимо определять, на какую дату мы задаем или получаем значение константы. Чтобы прочитать значение константы, используется функция `Получить()`; чтобы установить значение константы — функция `Установить()`. Примеры приведены в листинге 4.2.

Листинг 4.2. Работа с периодическими константами

```
// Выводим минимальный размер месячной оплаты труда.
Сообщить(Константа.МРОТ.Получить('01.01.2005'));
// Устанавливаем минимальный размер месячной оплаты труда
Константа.МРОТ.Установить('01.01.2005',720);
Константа.МРОТ.Установить('01.09.2005',800);
Константа.МРОТ.Установить('01.05.2006',1100);
```

4.3. Справочники

Справочник — это средство для работы со списками однородных элементов данных. Название и структура каждого конкретного справочника определяются при его создании в конфигураторе. У любого справочника существуют два реквизита, которые создаются автоматически, — **Код** и **Наименование**. Реквизиты справочников могут быть периодическими, то есть иметь значения, связанные с датой. При изменении значения периодического реквизита старое значение сохраняется, при этом новое значение начинает действовать с указанной даты, старое — до указанной даты. Свойства справочника редактируются в окне редактирования **Справочник** (рис. 4.3).

Помимо системных реквизитов можно добавить произвольное количество реквизитов, присущих данному виду справочника. Например, для справочника "Валюты" такими реквизитами являются курс и кратность. Поскольку эти реквизиты являются периодическими, то на вкладке **Дополнительные** (рис. 4.4) установим флажок **Периодический**. Для визуального представления справочника существует несколько форм:

- "Форма элемента";
- "Форма группы";
- "Формы списка" (их может быть несколько).

После создания справочника "Валюты" в конфигурации можно зайти в программу в режиме "1С:Предприятие" и ввести список элементов в справочник "Валюты". Доступ к справочнику можно получить через системное меню **Операции**, пункт **Справочники**.

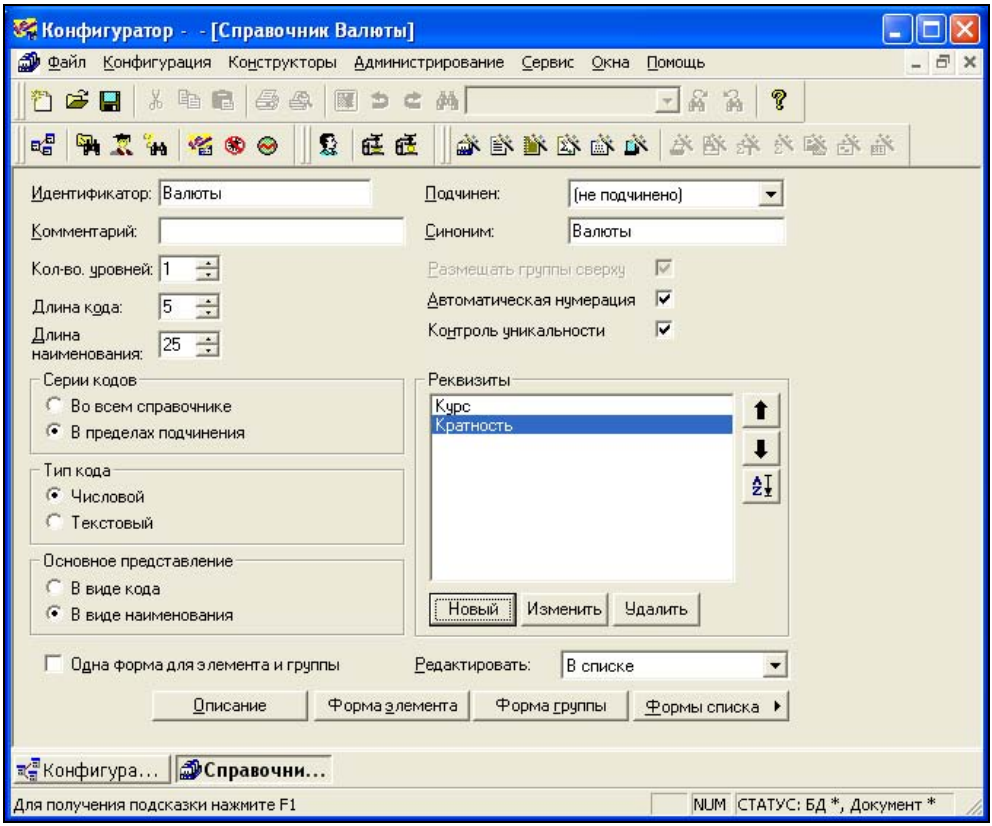


Рис. 4.3. Настройка справочника "Валюты"

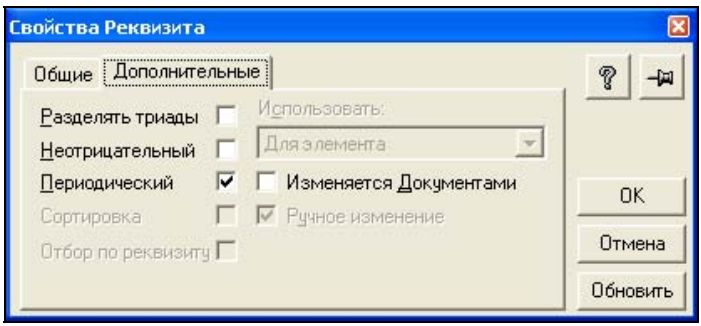


Рис. 4.4. Свойства реквизита "Курс" справочника "Валюты"

Для работы со справочником на встроенном языке существует агрегатный тип данных "Справочник".

Задание 4.1

Создайте следующие объекты метаданных.

1. Справочник "Заимодавцы" — сведения об организациях или физических лицах, предоставляющих свои средства в долг.
2. Справочник "Агенты" — сведения о структурных подразделениях или лицах, несущих ответственность за заключение и выполнение договора займа.
3. Справочник "Валюты" — сведения о валютах, имеющий *периодические* реквизиты "Курс" и "Кратность".
4. Константа `БазоваяВалюта` (тип "Справочник.Валюты").

4.3.1. Подчиненные справочники

Если один справочник подчинен другому (задается в поле **Подчинен** окна редактирования свойств справочника (см. рис. 4.3)), то каждый элемент подчиненного справочника будет соответствовать элементу справочника-владельца. Например, справочник "Договоры" в типовой конфигурации подчинен справочнику "Контрагенты" (рис. 4.5). Для организации подчинения у справочника есть специальный реквизит "Владелец", в котором хранится ссылка на элемент справочника-владельца.

Система поддерживает работу с подчиненными справочниками следующим образом. При открытии подчиненного справочника вы не увидите ни одного элемента до тех пор, пока не будет выбран элемент справочника-владельца. При определении элемента диалога, имеющего тип "Справочник", подчиненный некоторому справочнику-владельцу, в свойствах реквизита можно заполнить поле **Связан**, в котором указать имя реквизита, содержащего ссылку на элемент справочника-владельца.

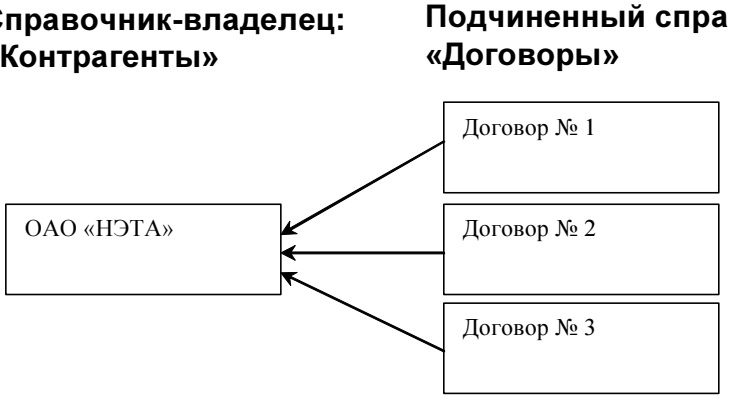


Рис. 4.5. Связь между справочниками

Вопрос о том, что лучше — использовать подчинение справочников или заводить реквизит типа ссылки на справочник, остается на усмотрение разработчика. Разработчики типовых конфигураций применяют оба подхода. На наш взгляд использование подчиненных справочников оправдано в следующих случаях:

- подчиненный справочник реализует табличную часть справочника-владельца (отметим, что в восьмой версии реализована поддержка табличных частей в справочниках);
- подчиненный справочник подчинен только одному справочнику-владельцу;
- нет необходимости просматривать весь список элементов подчиненного справочника без отбора по элементу справочника-владельца.

Использование реквизита для организации связи двух справочников является более гибким инструментом. Например, можно в качестве типа реквизита указать "Справочник", что означает, что в этот реквизит можно записать ссылку на справочник любого вида, предварительно назначив тип реквизита функцией `НазначитьТип()`.

4.3.2. Иерархия элементов

Справочник может иметь иерархическую структуру (рис. 4.6). Число уровней иерархии определяется в поле **Кол-во уровней** окна редактирования. Для задания иерархии используется реквизит "Родитель", который определяет, к какой группе относится элемент. Глубина вложенности групп настраивается в конфигураторе и *не может превышать 10 уровней*.

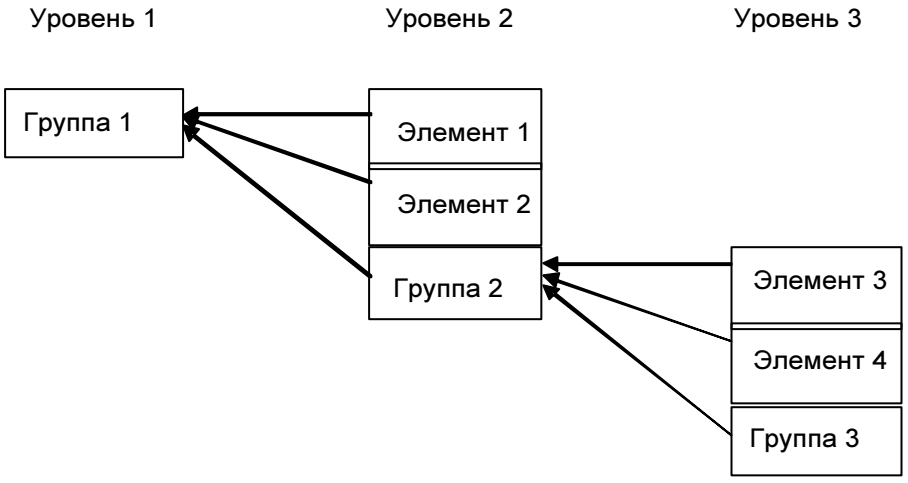


Рис. 4.6. Организация иерархической структуры

4.3.3. Работа со справочником на встроенном языке

С помощью переменной агрегатного типа "Справочник" можно выполнять различные действия с элементами справочника: создание новых элементов и групп, поиск элементов, удаление или пометка на удаление и др. Рассмотрим основные функции на примерах.

Ввод нового элемента справочника

После создания справочника валют желательно, чтобы базовой в нем была национальная валюта, относительно которой измеряются все остальные валюты. Добавить новый элемент справочника можно с помощью последовательного выполнения функций `Новый()` и `Записать()`, как показано в листинге 4.3.

Листинг 4.1. Добавление валюты "Рубль" в справочник "Валюты"

```
Спр = СоздатьОбъект("Справочник.Валюты");
Спр.Новый();
// Указываем, на какую дату будут задаваться периодические реквизиты
Спр.ИспользоватьДату('01.01.1980');
Спр.Код=810;
Спр.Наименование = "Рубль";
Спр.Курс=1;
Спр.Кратность=1;
Спр.Записать(); // Происходит запись нового элемента в базу данных
```

Запись элемента справочника выполняется в режиме транзакции *(см. главу 3)*. Поэтому может сложиться ситуация, когда при попытке записи будет выдана ошибка. Например, если в настройках справочника стоит флаг **Контроль уникальности** (см. рис. 4.3), то при записи элемента справочника проверяется, есть ли в информационной базе элемент с таким кодом или нет.

Замечания

1. Если в настройках справочника стоит флаг **Автоматическая нумерация**, то при создании нового элемента автоматически будет присваиваться код на единицу больший, чем максимальный. Если тип кода числовой, то проблем автонумерации нет. Если же код текстовый, то возможно возникновение следующей проблемы: два текстовых кода сравниваются посимвольно, начиная с самого первого символа слева. Если первые символы равны, то анализируются вторые символы и т. д. Например, код "09" будет меньше, чем код "1". При автоматической нумерации находится код с максимальным номером, и к нему прибавляется единица. Однако если максимальный номер равен "9" или "99" или "999" и т. д., то система считает, что увеличить этот код уже нельзя. По-

этому важно при автоматической нумерации оставлять ведущие нули кода, т. е. не заменять, например, код "00001" на "1". Если же нумерация все-таки сбилась, то необходимо выполнить перенумерацию кодов элементов вручную или программно.

2. Не используйте настройку справочника **Серии кодов в пределах подчинения**. Лучше использовать вариант **Серии кодов во всем справочнике**, чтобы впоследствии не было проблем при перемещении элементов из группы в группу или при выполнении различных операций по обмену данными.

Поиск элемента справочника

Если выполнить программу, представленную в листинге 4.3, второй раз, то будет выдана ошибка "Код не уникальный!". Как избежать данной ошибки? Самый простой вариант — перед записью элемента искать, есть ли элемент с таким кодом в информационной базе. Найти элемент можно одной из четырех функций:

- НайтиПоКоду()
- НайтиПоНаименованию()
- НайтиПоРеквизиту()
- НайтиЭлемент()

В случае поиска по реквизиту необходимо, чтобы элементы справочника были отсортированы по этому реквизиту. Для этого в свойстве реквизита на вкладке **Дополнительно** нужно поставить флажок **Сортировка**.

В листинге 4.4 приведена функция поиска базовой валюты по ее коду. При неудачном поиске создается новый элемент. Процедура возвращает ссылку на элемент справочника — базовую валюту.

Листинг 4.4. Функция поиска базовой валюты

```
Функция БазоваяВалюта()
Спр = СоздатьОбъект("Справочник.Валюты");
Если Спр.НайтиПоКоду(810)=0 Тогда
        Спр.Новый();
        // Указываем, на какую дату будут задаваться периодические
        реквизиты
        Спр.ИспользоватьДату('01.01.1980');
        Спр.Код=810;
        Спр.Наименование = "Рубль";
        Спр.Курс=1;
        Спр.Кратность=1;
        Спр.Записать(); // Происходит запись нового элемента в базу данных
```

```
КонецЕсли;
Возврат Спр.ТекущийЭлемент();
КонецПроцедуры
```

Получение ссылки на элемент справочника

Следующий вопрос, который мы поставим, — как определить какая валюта является базовой? Ведь, в общем случае, базовая валюта зависит от страны, в которой эксплуатируется программа. Для этого введем константу `БазоваяВалюта` типа "Справочник.Валюты" и инициализируем ее, как показано в листинге 4.5.

Листинг 4.5. Инициализация константы `БазоваяВалюта`

```
// Проверяем, задана ли константа БазоваяВалюта
Если Константа.БазоваяВалюта.выбран()=0 Тогда
        // Пользуемся функцией, описанной в листинге 4.4
        Константа.БазоваяВалюта=БазоваяВалюта();
КонецЕсли;
```

Отметим, что неправильной будет запись типа

```
Константа.БазоваяВалюта = "Рубль";
```

по причине того, что тип константы — ссылка на элемент справочника "Валюты", а присваивается значение типа "Строка". Результатом этой команды будет не ошибка, а присвоение константе пустого значения. Ссылку на элемент справочника мы получаем функцией `ТекущийЭлемент()`, как это показано в листинге 4.4.

Работа с группами справочника

В справочнике могут быть данные двух типов: "Элементы" и "Группы". Чтобы определить тип текущего элемента справочника, можно воспользоваться функцией `ЭтоГруппа()`, которая возвращает единицу, если текущий элемент справочника типа "Группа", и ноль — в противном случае.

Добавим в конфигурацию справочник "Агенты" с количеством уровней 2, и создадим в нем программно две группы — организации и физические лица, как показано в листинге 4.6. Ссылки на созданные группы запомним в соответствующих переменных.

Листинг 4.6. Создание групп справочника

```
Спр=СоздатьОбъект("Справочник.Агенты");
Спр.НоваяГруппа();
```

```
Спр.Наименование  = "Организации";
Спр.Записать();
ССылкаОрганизации=Спр.ТекущийЭлемент();
Спр.НоваяГруппа();
Спр.Наименование  = "Физические лица";
Спр.Записать();
ССылкаФизЛица=Спр.ТекущийЭлемент();
```

Теперь создадим агента "Иванов" и поместим его в группу "Физические лица", как показано в листинге 4.7.

Листинг 4.7. Привязка элемента справочника к группе

```
Спр.Новый();
Спр.Наименование = "Иванов";
Спр.Родитель=ССылкаФизЛица;
Спр.Записать();
```

Чтобы определить принадлежность текущего элемента какой-либо группе, используется функция `ПринадлежитГруппе(<Группа>)`. Чтобы определить уровень элемента в иерархии групп и элементов, используется функция `Уровень()`. Нумерация уровней начинается с единицы.

Обработка элементов справочника

Обычно для обработки элементов справочника используется следующая последовательность команд, приведенная в листинге 4.8.

Листинг 4.8. Обработка элементов справочника

```
Спр=СоздатьОбъект("Справочник.Агенты");
Спр.ИспользоватьДату(<Дата>); // Используется, если есть периодические
// реквизиты
Спр.ИспользоватьВладельца(<Владелец>); // Используется, если нужно
// выбрать только элементы, подчиненные Владельцу
Спр.ИспользоватьРодителя(<Группа>); // Используется, если нужно выбрать
// только элементы, принадлежащие Группе
Спр.ВыбратьЭлементы();        // Делаем выборку элементов
Пока Спр.ПолучитьЭлемент()=1 Цикл
        // Обработка текущего элемента выборки, например
        Сообщить(" "+Спр.Код+" "+Спр.Наименование);
КонецЦикла;
```

Удаление элементов справочника

Для удаления текущего элемента справочника используется функция `Удалить(<Режим>)`, где `<Режим>` может иметь значение 0 — пометка на удаление, 1 — непосредственное удаление (используется по умолчанию). Вообще непосредственное удаление задавать не рекомендуется из-за возможного нарушения ссылочной целостности базы данных.

Внимание!

Чтобы удалить помеченные на удаление объекты через систему 1С:Предприятие 7.7, нужно зайти в программу в монопольном режиме и в меню **Операции** выбрать пункт **Удаление помеченных объектов...**

4.4. Документы

Документы в системе 1С: Предприятие 7.7 используются для ввода, просмотра и корректировки информации о совершаемых хозяйственных операциях. Документ может быть электронным аналогом первичного "бумажного" документа, а также может использоваться для выполнения различных регламентных операций (типа закрытия месяца).

При разработке учетной системы на базе 1С:Предприятия 7.7 нужно четко понимать отличия документа от справочника, его особенности и возможности.

4.4.1. Отличия документа от справочника

Рассмотрим основные отличия документа от справочника:

- документ имеет *позицию* на временной оси, которая состоит из даты и времени с точностью до секунды, плюс некоторые дополнительные данные, определяющие взаимное расположение документов в пределах одной секунды. Это означает, что для любых двух документов можно сказать, какой из них находится на временной оси раньше, а какой позже. Элементы справочника по времени не упорядочены;
- документ может *проводиться* — формировать проводки по счетам бухгалтерского учета, движения по регистрам оперативного учета, записи журнала расчетов. Если документ проведен, то он влияет на бухгалтерские или оперативные итоги, периодические реквизиты справочников или записи журнала расчетов. Если документ не проведен, то это означает, что он находится в работе, а именно: или еще не полностью введена информация, или факт, который документ отражает, еще не свершился;

- документы и их движения (бухгалтерские проводки, записи регистров и журналов расчетов) могут иметь единую *последовательность*. Это означает, что корректность сформированных документом движений существенно зависит от последовательности проведения документов. В самом деле, если задним числом удалить или изменить приход товара на склад, то последующий расход может оказаться некорректным (например, остаток товара на складе окажется меньше нуля!). Важной характеристикой последовательности является ее граница — позиция на временной оси, которая показывает, что с этого момента документы последовательности должны быть перепроведены *(см. главу 10)*;
- документ имеет одну табличную часть, в то время как справочник табличной части не имеет (данное различие устранено в восьмой версии 1С:Предприятия).

Чтобы добавить новый вид документа, нужно открыть конфигурацию, навести курсор мыши на раздел **Документы**, нажать правую кнопку мыши и выбрать пункт **Новый Документ**. Параметры вида документа задаются в форме, представленной на рис. 4.7.

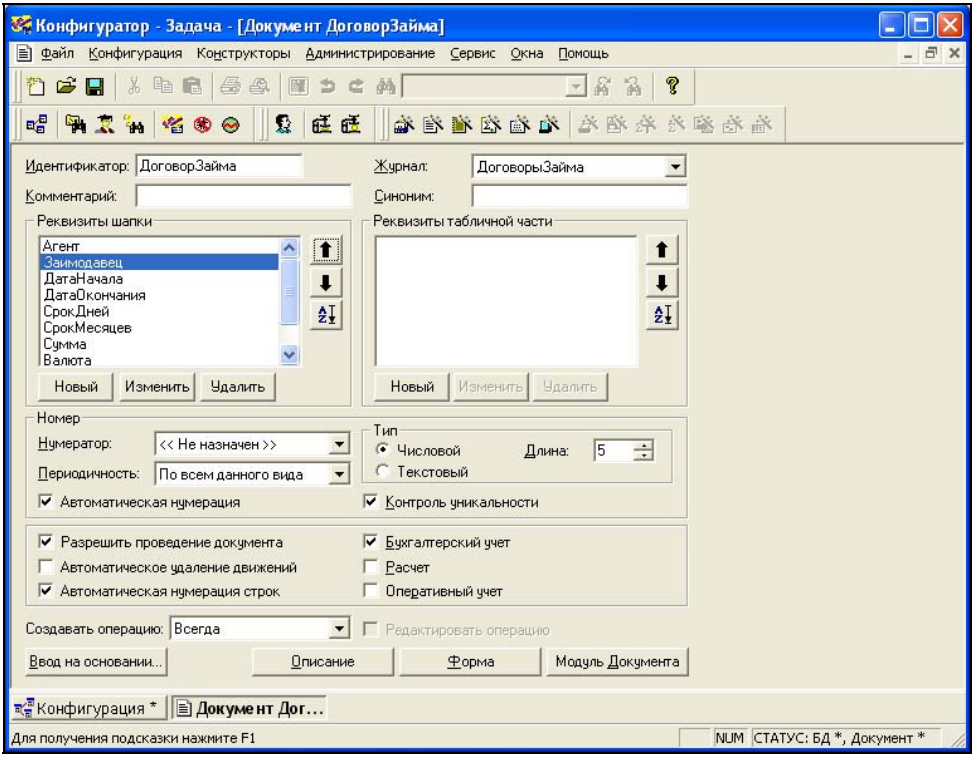


Рис. 4.7. Форма настройки документа

Форма документа редактируется при нажатии кнопки **Форма**. Для размещения реквизитов документа на форме нужно выбрать пункт **Реквизиты** из меню **Вставить**.

4.4.2. Реквизиты документа

У любого документа есть два системных реквизита `ДатаДок` и `НомерДок`. Кроме системных реквизитов программист может добавить произвольное количество собственных реквизитов двух видов: реквизиты шапки и реквизиты табличной части. У реквизита обязательно должен быть задан тип. В свойствах реквизита табличной части на вкладке **Дополнительно** может быть установлен флажок **Итог по колонке**, для поддержки автоматического вычисления и хранения суммы, с последующим отображением в графе журнала документов.

Если один и тот же реквизит должен присутствовать во всех документах конфигурации (например, автор документа, комментарий и др.), то в системе 1С:Предприятие 7.77 можно определить его один раз в разделе общих реквизитов (раздел метаданных **Документы**, подраздел **Общие реквизиты**).

Задание 4.2

1. Создайте документ `"ДоговорЗайма"`, имеющий следующие реквизиты:
   - номер договора (системный реквизит `НомерДок`);
   - дата заключения договора (системный реквизит `ДатаДок`);
   - `Агент` (тип `"Справочник.Агенты"`);
   - `Заимодавец` (тип `"Справочник.Заимодавцы"`);
   - `ДатаНачала` (тип `"Дата"`) — дата начала действия договора;
   - `Сумма` (тип `"Число"`) — сумма договора;
   - `Валюта` (тип `"Справочник.Валюты"`) — валюта договора;
   - `СрокМесяцев` (тип `"Число"`) — число полных месяцев;
   - `СрокДней` (тип `"Число"`) — число дней в неполном месяце;
   - `ДатаОкончания` (тип `"Дата"`) — дата окончания договора;
   - `ПроцентСрочный` (тип `"Число"`) — процент годовых, выплачиваемый заимодавцу при возврате денежных средств по окончании срока договора;
   - `ПроцентДосрочный` (тип `"Число"`) — процент годовых, выплачиваемый заимодавцу при возврате денежных средств при досрочном завершении договора.

2. Откройте форму документа и вставьте реквизиты документа в форму диалога.

Указание

Для добавления реквизитов документа в форму диалога используйте пункт **Реквизиты** меню **Вставить**.

3. Обеспечьте автоматический расчет даты окончания договора на основании даты начала и срока договора.

4.4.3. Проведение документа

Характерной особенностью документа является возможность его проведения. Для этого при настройке документа необходимо поставить флажок **Разрешить проведение документа**. Далее отметить, по каким компонентам документ будет формировать движения. Обратите внимание: флажок **Бухгалтерский учет** можно поставить только в том случае, если в конфигурации есть хотя бы один план счетов. Формирование движений происходит в **Модуле документа** в предопределенной процедуре `ОбработкаПроведения()`. Примеры формирования движений документа мы будем рассматривать в главах, посвященных соответствующим компонентам.

Проведение документа происходит:

- *интерактивно* при нажатии кнопки, у которой на вкладке **Дополнительно** задана формула `#Провести`;
- *программно*, с помощью функции `Провести()`;
- при групповом проведении документов (в режиме "1С:Предприятие" меню **Операции**, пункт **Проведение документов**).

4.4.4. Работа с документом на встроенном языке

Для работы с документами предназначен агрегатный тип данных "Документ", который предоставляет все основные функции, доступные интерактивно: ввод нового документа, обработка документов, поиск, удаление, проведение и др.

Ввод нового документа

Для ввода документа используется функция `Новый()`, далее заполняются реквизиты шапки документа. Для ввода новой строки документа используется функция `НоваяСтрока()` и заполняются реквизиты табличной части документа. Запись документа в информационную базу производится функцией `Записать()`.

Обработка документов

Обычно для обработки документов используется последовательность команд, приведенная в листинге 4.9.

Листинг 4.9. Обработка документов

```
Док=СоздатьОбъект("Документ.ВидДокумента");
Док.ВыбратьДокументы(); // Открываем выборку документов
Пока Док.ПолучитьДокумент()=1 Цикл
        // Обработка шапки документа, например
        Сообщить("Вид документа: "+Строка(Док.Вид()));
        Сообщить("Дата документа: "+Строка(Док.ДатаДок));
        Сообщить("Номер документа: "+Строка(Док.НомерДок));
        Док.ВыбратьСтроки();// Делаем выборку табличной части документа
        Пока Док.ПолучитьСтроку()=1 Цикл
                // Обработка строк табличной части документа, например
                Сообщить("Товар: "+Док.Товар.Наименование);
        КонецЦикла
КонецЦикла;
```

Задание 4.3

1. Создайте внешний отчет, выводящий в окно сообщений список всех договоров займа, заключенных в заданный период. В форме диалога должны задаваться дата начала периода, дата окончания периода, заимодавец (если не выбран, то выводить все договоры), агент (если не выбран, то выводить все договоры). В окно сообщений должны выводиться: номер договора, дата начала договора, дата завершения договора, срок договора, сумма, валюта, процент срочный, процент досрочный.
2. Создайте внешнюю обработку, изменяющую все договоры займа по следующему алгоритму: процент договора досрочный = процент договора срочный/2.
3. Напишите обработку, создающую программным образом:
   - новый элемент справочника "Заимодавцы" с наименованием "Сбербанк РФ";
   - новый элемент справочника "Агенты" с наименованием "Иванов И.И.";
   - новый договор займа, заключенный со Сбербанком агентом Ивановым на сумму $1000 на период с 01.01.2005 по 31.10.2005.

При вводе новых элементов сделайте проверку на существование элементов с таким же наименованием.

4.4.5. Журналы документов

Для просмотра списка документов используются "Журналы документов". Создание журнала производится в конфигураторе. Бывают журналы трех видов:

- обычные;
- дополнительные;
- общие.

Один вид документа может находиться в одном обычном журнале и в произвольном количестве дополнительных журналов. Общий журнал содержит документы всех видов и обладает наибольшими возможностями по отбору документов.

В диалоговой форме задаются графы — реквизиты документов, входящих в журнал. Реквизиты табличной части числового типа можно добавить только в том случае, если в свойствах реквизита на вкладке **Дополнительно** стоит флажок **Итог по колонке**.

Задание 4.4

Создайте журнал документов `"ДоговорыЗайма"`, включающий графы: дата начала, дата окончания, номер, заимодавец, агент, сумма, валюта.

4.4.6. Графы отбора

При просмотре журнала документов часто возникает желание отобрать список документов по какому-либо критерию. Такие критерии в системе 1С:Предприятие 7.7 задаются в виде *граф отбора*.

Чтобы создать новую графу отбора, нужно открыть раздел метаданных **Журналы** и выбрать подраздел **Графы отбора**. Далее нажать правую кнопку мыши и выбрать пункт **Новая Графа**. В форме настройки графы отбора (рис. 4.8) нужно задать реквизиты документов, по которым будет производиться отбор.

Внимание!

Отбор можно делать только в общих журналах.

Установка отбора выполняется либо по нажатию соответствующей кнопки на панели инструментов общего журнала, либо программно в модуле формы журнала документов с помощью функции `УстановитьОтбор(<ИмяОтбора>, <ЗначениеОтбора>)`, где `<ИмяОтбора>` — название отбора, как оно задано в конфигураторе, а `<ЗначениеОтбора>` — значение соответствующего типа.

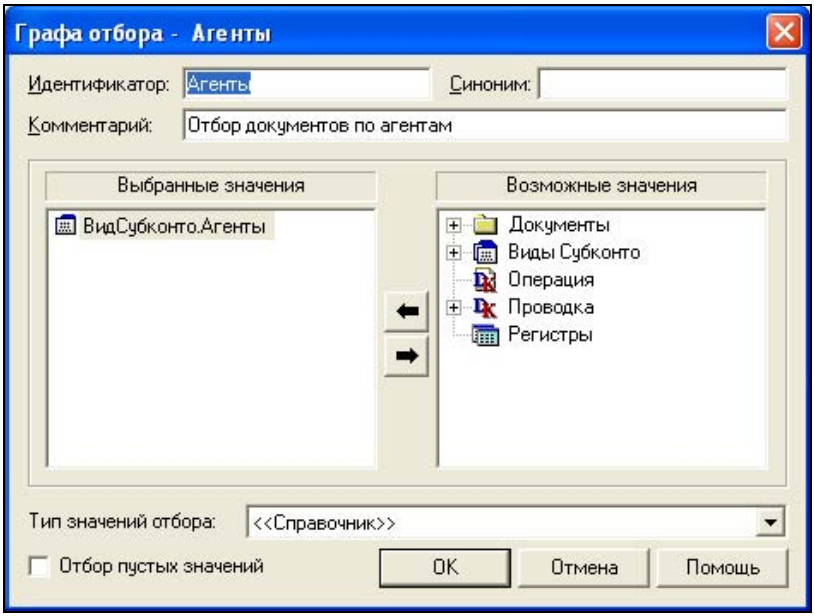


Рис. 4.8. Настройка графы отбора

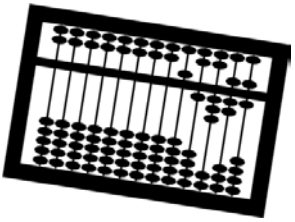
Замечание

Отбор также можно делать по виду и общим реквизитам документов. Однако отобрать документы одновременно по нескольким критериям невозможно. Искусственно это можно сделать, создав вспомогательный реквизит документа, в котором хранить значение отбора. Например, чтобы отобрать документы перемещения по складу-источнику и складу-приемнику, нужно создать новый реквизит документа "ИсточникПриемник", например, строкового типа, и при сохранении документа записывать туда названия склада-источника и склада-приемника.

4.5. Контрольные вопросы

1. Чем базовые объекты отличаются от прикладных?
2. Как задается значение константы? Может ли оно изменяться?
3. Может ли на одну дату быть задано два разных значения периодической константы, периодического реквизита справочника?
4. Чем иерархический справочник отличается от линейного? Какова максимальная глубина вложенности иерархического справочника? Какой элемент является родителем для всех элементов первого уровня?

5. Как можно связать два справочника?
6. Как получить ссылку на элемент справочника, документ?
7. Чем отличаются справочники и документы?
8. Чем отличаются реквизиты шапки документа от реквизитов табличной части?
9. Что такое "проведение документа"? Какие флажки необходимо установить в форме настройки документа, чтобы он мог проводиться?
10. Какие виды журналов документов бывают, чем они отличаются?
11. Что такое "графа отбора", зачем она используется?



Глава 5

Механизм запросов

Система 1С:Предприятие 7.7 предоставляет два основных способа извлечения информации из информационной базы:

- с помощью методов объектов метаданных, таких как константы, справочники, документы и т. д. Примеры получения констант, данных об элементах справочников и документах, мы рассмотрели в *главе 4*;
- с помощью механизма универсального запроса можно извлечь данные почти всех объектов системы 1С:Предприятие: справочников, документов, регистров, бухгалтерских итогов и др.

Последовательность действий при работе запросом следующая:

1. Создаем переменную агрегатного типа "Запрос".
2. В переменной типа "Строка" описываем текст запроса на языке запросов.
3. Выполняем запрос.
4. Обрабатываем результат запроса.

5.1. Консоль запросов

Для изучения языка запросов нам понадобится обработка **КонсольЗапросов** (рис. 5.1). Для ее создания выполните следующие действия в конфигураторе:

1. Создайте новый внешний отчет (обработку).
2. Поместите на форму диалога реквизит диалога `ТекстЗапроса`, тип "Строка", неограниченной длины, многострочный. В свойствах реквизита **ТекстЗапроса** отметьте также флажок **Сохранять при сохранении настройки**.
3. Поместите на форму диалога таблицу значений `РезультатЗапроса`.
4. В модуле формы напишите процедуру `Сформировать()`, как показано в листинге 5.1.
5. Сохраните обработку с именем `"КонсольЗапроса"`.

С помощью этой обработки можно набрать текст запроса и посмотреть его результат.

Листинг 5.1. Модуль формы обработки "КонсольЗапросов"

```
Процедура Сформировать()
Запрос=СоздатьОбъект("Запрос");
Если Запрос.Выполнить(ТекстЗапроса)=1 Тогда
        // Выгружаем результат запроса в таблицу значений
        Запрос.Выгрузить(РезультатЗапроса);
КонецЕсли;
КонецПроцедуры
```

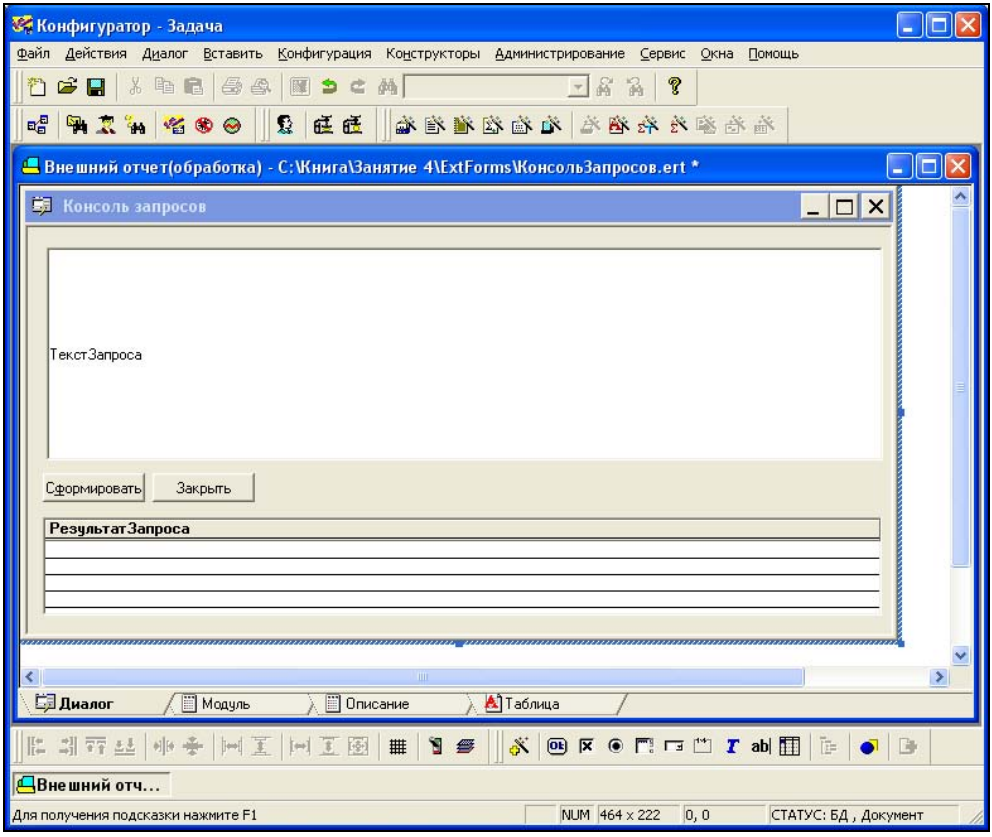


Рис. 5.1. Обработка **КонсольЗапросов**

5.2. Язык запросов

На языке запросов мы описываем, данные каких объектов мы хотим извлечь и выполнить над ними некоторые действия, такие как группировка, упорядочивание, вычисление функций. При этом мы можем указать период, за который выбираются данные, наложить дополнительные условия на выбираемые данные.

Текст запроса состоит из операторов. Каждый оператор заканчивается точкой с запятой. В тексте запроса могут быть комментарии, которые начинаются с "//" и заканчиваются концом строки. Операторы могут записываться в любом порядке, однако следует помнить, что интерпретатор языка запросов однопроходный, и, например, чтобы использовать внутреннюю переменную в операторах `Группировка`, `Функция` или `Условие`, ее предварительно нужно описать.

Задание 5.1

Подготовьте информационную базу для демонстрации работы с запросами. За основу возьмите конфигурацию из *главы 4*.

1. Введите в справочник `Агенты` элементы `Иванов И.И.` (код 1), `Петров П.П.` (код 2), `Александров А.А.` (код 3).
2. Введите в справочник `Заимодавцы` элементы `Сбербанк РФ` (код 1), `Банк КМБ` (код 2).
3. Введите документы `ДоговорЗайма` согласно заданию 2.3.

5.2.1. Внутренние переменные запроса

Внутренняя переменная — это переменная, объявленная в тексте описания запроса следующим образом:

```
<ИмяПеременной> = <ОписаниеПеременной>;
```

где описание переменной указывает на доступный в языке запросов атрибут документа, справочника, регистра или журнала расчетов. Для получения ссылки на сам объект используется атрибут `ТекущийДокумент` для документов и `ТекущийЭлемент` — для элементов. Например, описание

```
Агент = Справочник.Агенты.ТекущийЭлемент;
```

означает, что во внутреннюю переменную `Агент` будут попадать ссылки на элементы справочника агентов. Описание

```
Агент = Документ.ДоговорЗайма.Агент;
```

также выбирает ссылки на агентов, но только тех, которые присутствуют в документах `ДоговорЗайма`.

Одна внутренняя переменная может иметь несколько описаний, разделенных запятыми, например,

```
Агент = Документ.ДоговорЗайма.Агент, Справочник.Агенты.ТекущийЭлемент;
```

Типы данных по всем описаниям должны совпадать, иначе произойдет ошибка выполнения запроса. При этом допускается ссылка на документы (справочники) разных видов. В этом случае внутренняя переменная получит тип

```
"Документ (справочник) неопределенного вида".
```

Несколько описаний внутренней переменной означает, что при ее использовании в качестве группировки будут выбраны все объекты, встречающиеся хотя бы в одном из описаний.

5.2.2. Группировки

Группировки запроса устанавливают порядок выбора информации. Группировка описывается следующим образом:

```
Группировка <ИмяГруппировки>
[Упорядочить по <Порядок>][Без Упорядочивания]
[Без Групп]
[Все [ВошедшиеВЗапрос]];
```

В качестве имени группировки может быть выбрана одна из внутренних переменных или предопределенная группировка:

- `Документ`
- `СтрокаДокумента`
- `День`
- `Неделя`
- `Месяц`
- `Квартал`
- `Год`

Внимание!

Отметим, что если в тексте запроса не описана ни одна группировка, то результат запроса будет пустой, при этом запрос выполнится без ошибок.

Давайте откроем обработку **КонсольЗапросов** в режиме "1С:Предприятие" и выполним текст запроса, представленный в листинге 5.2.

Листинг 5.2. Текст запроса по справочнику агентов

```
Агент = Справочник.Агенты.ТекущийЭлемент;
Группировка Агент;
```

Результат запроса приведен на рис. 5.2. Как мы видим, результат содержит всех агентов и одну пустую строку вначале — это строка итогов по всем группировкам. Чтобы отключить итоги по группировкам, можно добавить в запросе оператор `БезИтогов`.

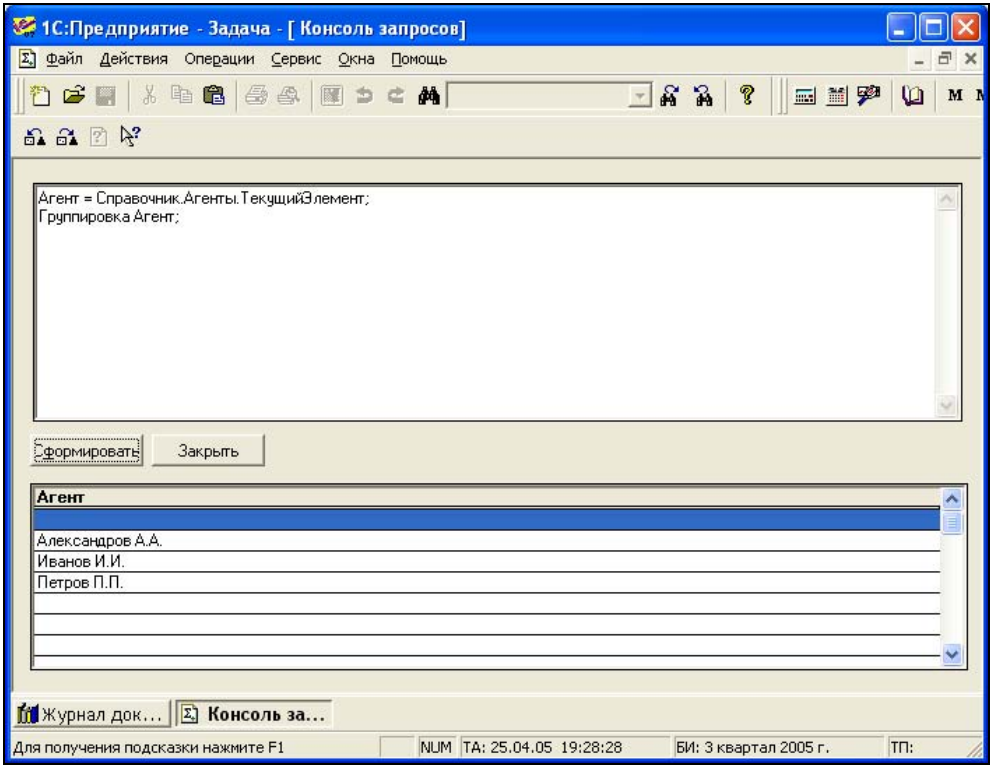


Рис. 5.2. Результат запроса по справочнику агентов

По умолчанию строки группировки упорядочиваются следующим образом: документы — по позиции на оси времени, справочники — по коду или наименованию в зависимости от основного представления. Чтобы упорядочить строки группировки иначе, нужно использовать в описании фразу `Упорядочить по` и указать значение упорядочивания. Например, чтобы список агентов в запросе листинга 5.2 выдавался в порядке кодов, можно написать так, как показано в листинге 5.3.

Листинг 5.3. Текст запроса по справочнику агентов с упорядочиванием по кодам

```
Агент = Справочник.Агенты.ТекущийЭлемент;
Группировка Агент Упорядочить по Агент.Код;
```

Для упорядочивания группировок могут также выступать функции запроса. Упорядочивание по группировке можно отключить, добавив ключевое слово `Без Упорядочивания`.

Для группировок по элементам с иерархической структурой можно отключить включение в результат запроса групп элементов с помощью ключевого слова `Без Групп`.

Если группировка построена на основе внутренней переменной типа "Справочник", то при использовании ключевого слова `Все` в результат запроса будут включены все элементы этого справочника.

Для предопределенных временных группировок при использовании ключевого слова `Все` подразумевается, что в запрос будут включены любые значения данных (в том числе нулевые) в каждый заданный момент времени с даты начала запроса по дату конца запроса (интервал задается оператором запроса `Период С...`).

Группировок в запросе может быть несколько. Каждая последующая группировка детализирует предыдущую, поэтому порядок группировок принципиально важен.

5.2.3. Условия

В запросе условия на отбираемые данные можно задать следующими операторами:

- `Обрабатывать[ПомеченныеНаУдаление|НеПомеченныеНаУдаление|Все];` — оператор уточняет, являются ли отбираемые в запросе объекты помеченными на удаление или нет. По умолчанию обрабатываются все объекты;
- `ОбрабатыватьДокументы[Непроведенные|Проведенные|Все];` — оператор уточняет, являются ли отбираемые в запросе документы проведенными или нет;

  Внимание!

  По умолчанию в запрос входят *только проведенные документы.*

- `[Период] С <Дата>|<ВнешняяПерем> [ПО <Дата>|<ВнешПеременная>];` — оператор уточняет, за какой период будут извлекаться данные документов и объектов, имеющих привязку по оси времени: бухгалтерских операций, регистров, записей журнала расчетов. В качестве даты может указываться дата, внешняя переменная или выражение на встроенном языке, заключенное в круглые скобки. Для документов отбор производится по реквизиту `ДатаДок`;

Внимание!

По умолчанию период запроса устанавливается в точку актуальности *(см. разд. 10.1.3)*, если установлена компонента "Оперативный учет", или на рабочую дату.

- `Условие(<ЛогическоеВыражение>);` — в этом операторе можно задать произвольное условие. Если в тексте запроса есть несколько операторов `Условие()`, то они объединяются логическим `И`.

Дополнительно к стандартным логическим операциям, определенным во встроенном языке *(см. разд. 3.9)*, в запросе может использоваться проверка вхождения элемента справочника в группу элементов или в список значений — операция `"в"`. При этом если группа задана пустым значением, то условие будет истинным для всех элементов справочника. Например, если группа агентов задана в переменной `ВыбАгент`, то запрос по договорам займа будет выглядеть так, как показано в листинге 5.4. Если попробовать выполнить этот запрос в обработке **КонсольЗапросов**, то будут выданы сообщения об ошибке, что не определены переменные `НачДата`, `КонДата` и `ВыбАгент`. Поэтому добавим на форму диалога реквизиты диалога `НачДата`, `КонДата` (меню **Вставить**, пункт **Реквизит диалога**, подпункт **Выбор периода**), поле для ввода `ВыбАгент` (тип "Справочник.Агенты").

Результат выполнения запроса приведен на рис 5.3.

Совет

Чтобы не заполнять каждый раз одни и те же значения в форме диалога, в панели инструментов формы есть две кнопки: **Открыть настройку** и **Сохранить настройку**. После того как все параметры формы введены (например, в форме отчета "КонсольЗапросов" это реквизит `ТекстЗапроса`), нажимаем кнопку **Сохранить настройку** и задаем название настройки. Чтобы вернуться к сохраненным настройкам, нажимаем кнопку **Открыть настройку** и выбираем интересующую нас настройку.

Листинг 5.4. Текст запроса по договорам займа с отбором по группе агентов

```
Период С НачДата По КонДата;
ОбрабатыватьДокументы все;
Агент = Документ.ДоговорЗайма.Агент;
Договор = Документ.ДоговорЗайма.ТекущийДокумент;
Группировка Договор;
Условие(Агент в ВыбАгент);
```

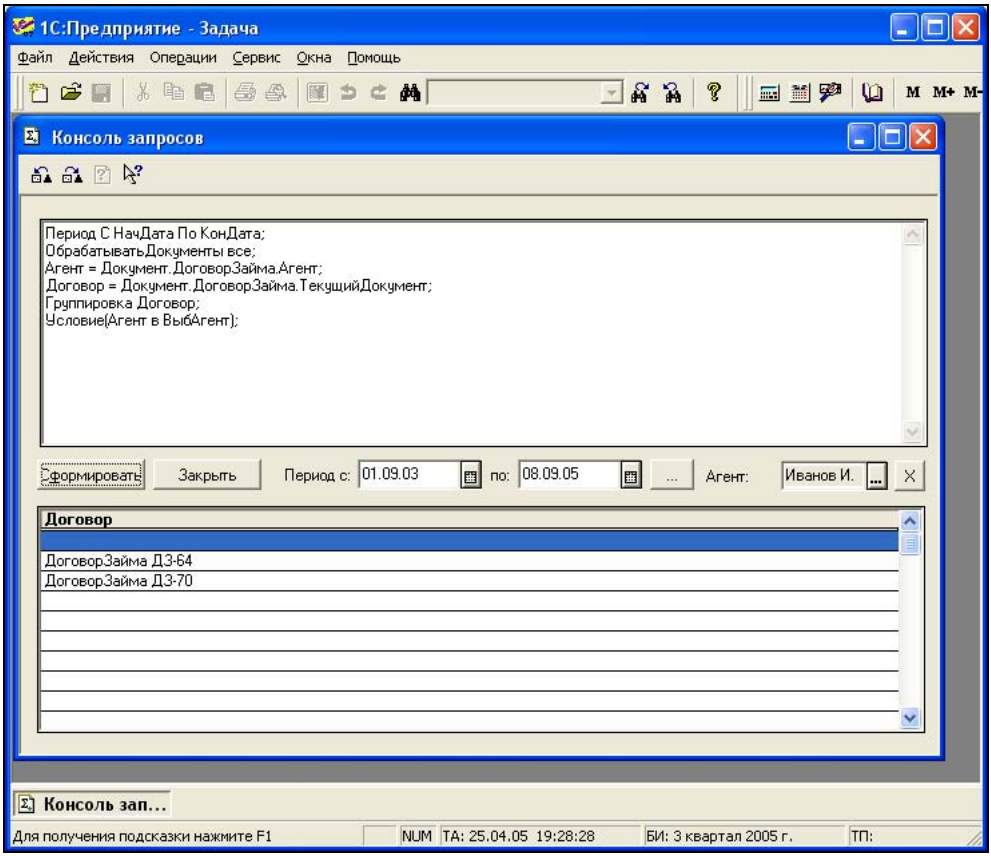


Рис. 5.3. Запрос по договорам займа с условием по периоду и группе агентов

Задание 5.2

Напишите запрос по договорам займа с отбором по группе агентов и группе заимодавцев.

5.2.4. Функции

До сих пор мы получали только значения группировок. Однако запрос еще умеет выполнять несложные вычисления, например, отвечать на вопросы типа "На какую сумму заключены договоры агентом Ивановым?". Для этого в языке запросов предусмотрен оператор `Функция`, имеющий следующий формат:

```
Функция <ИмяФункции> = <ТипФункции> (<Параметр>)[Когда (<Условие>)];
```

Здесь `<ТипФункции>` — одна из предопределенных функций:

- `Сумма`
- `Среднее`
- `Минимум`
- `Максимум`
- `Счётчик`

В качестве параметра функции может быть имя внутренней переменной типа "Число", а для функций `Сумма`, `Среднее`, `Минимум`, `Максимум` может быть арифметическое выражение в терминах встроенного языка 1С:Предприятия.

Сумма

Например, чтобы ответить на вопрос "На какую сумму заключены договоры агентом Ивановым?", можно предложить текст запроса, приведенный в листинге 5.5. Результат запроса приведен на рис. 5.4. Первой строкой в таблице идет итоговая сумма по всем значениям группировок.

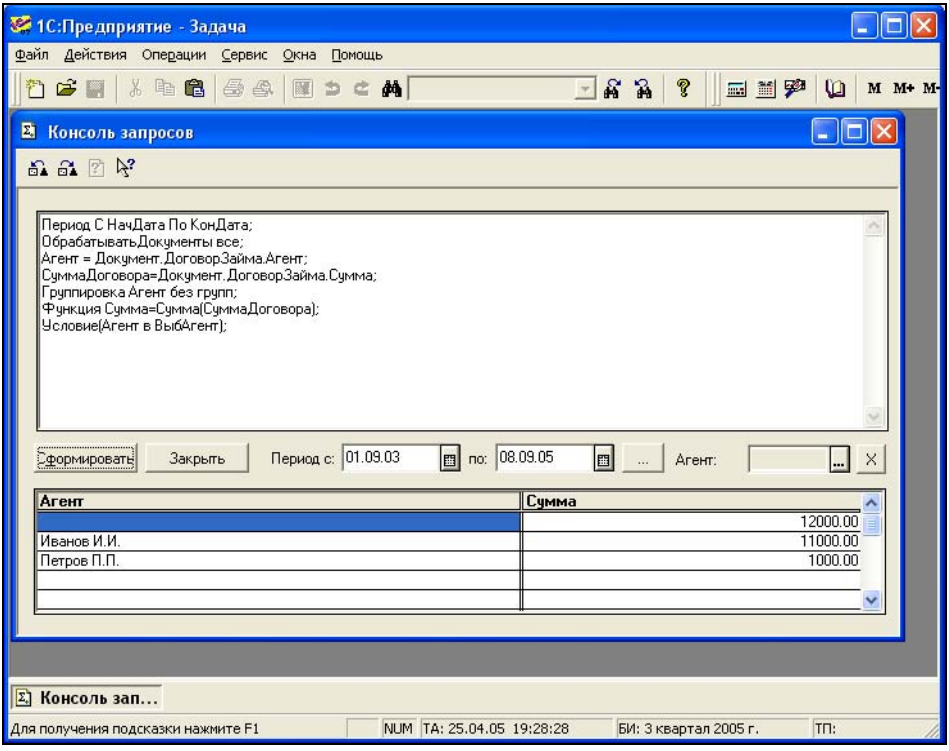


Рис. 5.4. Результат запроса, дающего список агентов и сумму заключенных ими договоров за период

Листинг 5.5. Текст запроса, дающего список агентов и сумму их договоров за период

```
Период С НачДата По КонДата;

ОбрабатыватьДокументы все;

Агент = Документ.ДоговорЗайма.Агент;

СуммаДоговора=Документ.ДоговорЗайма.Сумма;

Группировка Агент без групп;

Функция Сумма=Сумма(СуммаДоговора);

Условие(Агент в ВыбАгент);
```

Если внимательно посмотреть на результат, приведенный на рис. 5.4, то можно заметить некоторую некорректность: в сумму договоров, заключенных агентом, входят суммы в разных валютах. Чтобы исправить данную некорректность, добавим перевод суммы договора в базовую валюту по курсу (листинг 5.6). Кроме того, мы добавили в запрос еще одну дополнительную группировку — по договорам займа. Результат выполнения запроса приведен на рис. 5.5.

Листинг 5.6. Текст запроса по договорам с переводом суммы в базовую валюту

```
Период С НачДата По КонДата;

ОбрабатыватьДокументы все;

Агент = Документ.ДоговорЗайма.Агент;

СуммаДоговора=Документ.ДоговорЗайма.Сумма;

Договор=Документ.ДоговорЗайма.ТекущийДокумент;

Курс=Документ.ДоговорЗайма.Валюта.Курс;

Группировка Агент без групп;

Группировка Договор;

Функция Сумма=Сумма(СуммаДоговора*Курс);

Условие(Агент в ВыбАгент);
```

Замечание

Реквизит `Курс` справочника `Валюты` является периодическим. *Значения периодических реквизитов извлекаются запросом на конец периода запроса*. Историю периодического реквизита запросом получить невозможно!

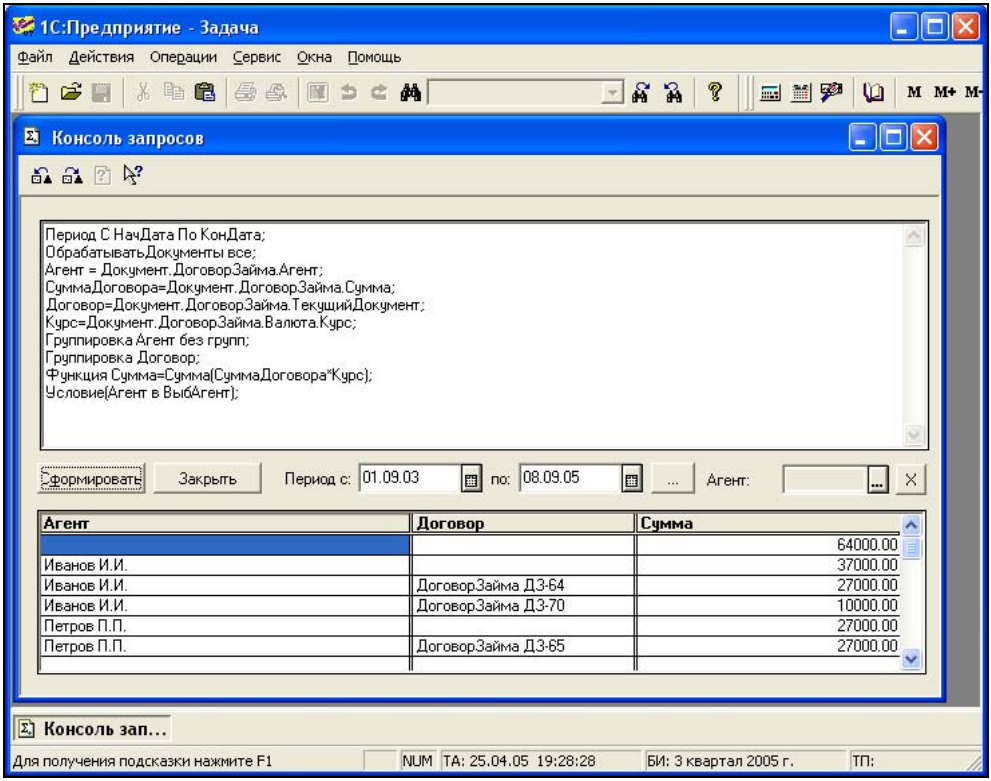


Рис. 5.5. Результат запроса по договорам займа с переводом суммы договора в базовую валюту

Счётчик

Функция `Счётчик` подсчитывает количество записей, вошедших в запрос.

Внимание!

Слово `Счётчик` пишется через букву "ё".

Например, чтобы определить, сколько договоров заключил каждый агент, мы можем написать текст запроса, приведенный в листинге 5.7. Результат запроса приведен на рис. 5.6.

Листинг 5.7. Текст запроса, определяющего количество договоров, заключенных агентами

```
Период С НачДата По КонДата;
ОбрабатыватьДокументы все;
```

```
Агент = Документ.ДоговорЗайма.Агент;
Группировка Агент без групп;
Функция КоличествоДоговоров=Счётчик();
```

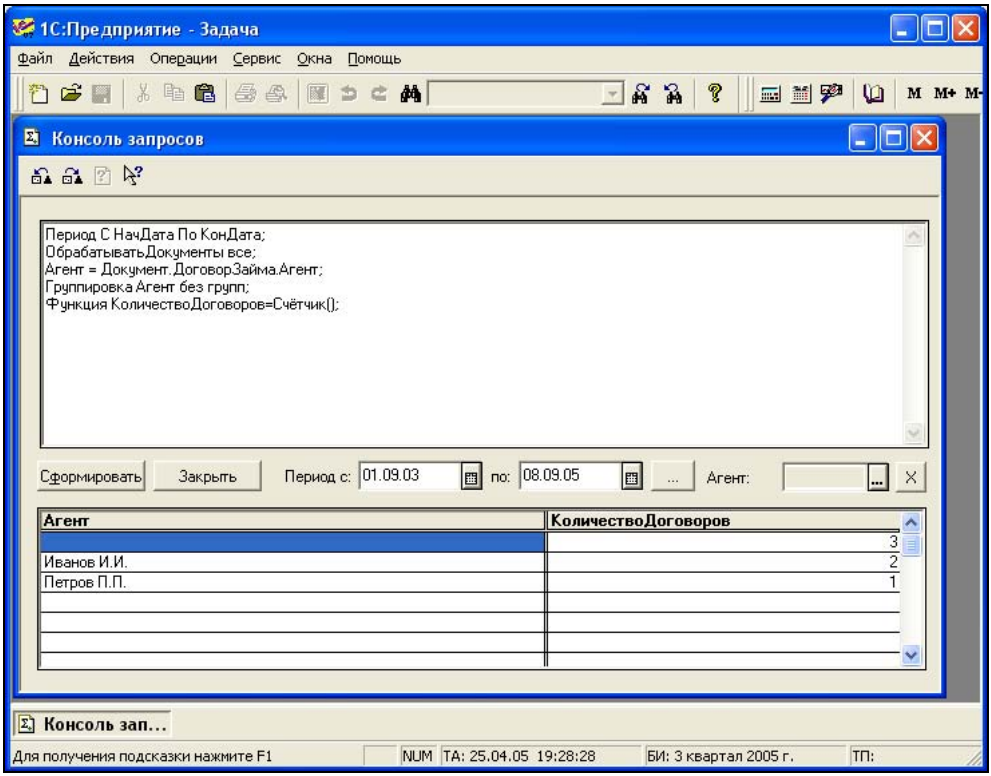


Рис. 5.6. Результат запроса, определяющего количество договоров, заключенных каждым агентом

Среднее, максимум, минимум

Функции `Среднее`, `Максимум` и `Минимум` определяют, соответственно, среднее, максимальное и минимальное значение по группировке. Подытоживая работу с функциями запроса, приведем текст запроса, использующий все перечисленные функции (листинг 5.8).

Листинг 5.8. Текст запроса, выдающего сводную информацию по агентам

```
Период С НачДата По КонДата;
ОбрабатыватьДокументы все;
Агент = Документ.ДоговорЗайма.Агент;
```

```
СуммаДоговора=Документ.ДоговорЗайма.Сумма;
Курс=Документ.ДоговорЗайма.Валюта.Курс;
Группировка Агент без групп;
Функция ОбщаяСумма=Сумма(СуммаДоговора*Курс);
Функция КолДоговоров=Счётчик();
Функция СредняяСумма=Среднее(СуммаДоговора*Курс);
Функция МаксСумма=Максимум(СуммаДоговора*Курс);
Функция МинСумма=Минимум(СуммаДоговора*Курс);
```

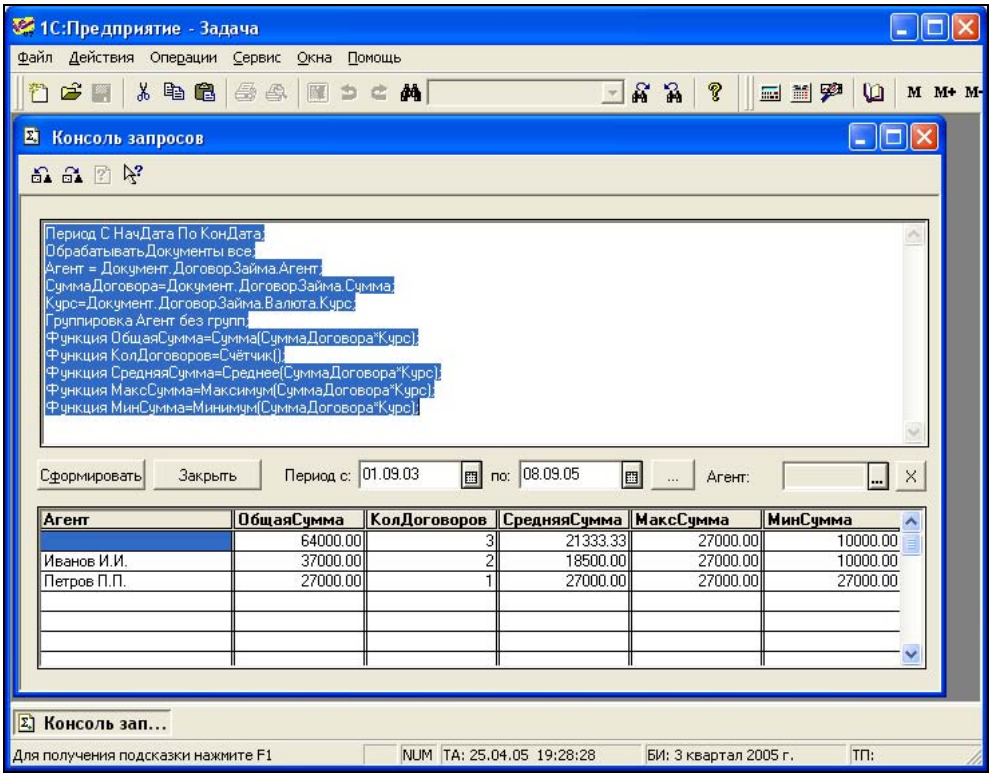


Рис. 5.7. Сводная информация по агентам

Упорядочивание по функциям

Значения функций можно использовать для упорядочивания группировок. Например, можно упорядочить список агентов по возрастанию общей суммы договоров, как показано в листинге 5.9.

Листинг 5.9. Текст запроса, дающего список агентов, упорядоченный по возрастанию общей суммы договоров

```
Период С НачДата По КонДата;
ОбрабатыватьДокументы все;
Агент = Документ.ДоговорЗайма.Агент;
СуммаДоговора=Документ.ДоговорЗайма.Сумма;
Курс=Документ.ДоговорЗайма.Валюта.Курс;
Функция ОбщаяСумма=Сумма(СуммаДоговора*Курс);
Группировка Агент без групп Упорядочить по ОбщаяСумма;
```

5.2.5. Язык запросов 1С и SQL

Читателям, знакомым с языком структурированных запросов SQL, язык запросов, реализованный в системе 1С:Предприятие 7.7, покажется достаточно бедным. Действительно, многие возможности SQL здесь не используются, некоторые механизмы запроса не всегда очевидны (например, связывание таблиц).

Замечание

Отметим, что в восьмой версии системы 1С:Предприятие язык запросов реализован в соответствии со стандартом SQL.

Если с помощью запроса не удается получить требуемую информацию, то порядок действий следующий: с помощью одного или нескольких запросов выбирается необходимая информация, которая затем выгружается в таблицы значений; дальнейшая обработка таблиц значений (сортировка, свертка, добавление новых колонок и вычисление значений в них) позволяет получить необходимую информацию.

5.2.6. Оптимизация запросов

Выборку одной и той же информации можно описать различными способами, при этом скорость выполнения запросов будет различаться на несколько порядков (естественно, в зависимости от объема базы данных).

Замечание

Использование запросов наиболее эффективно при работе системы 1С:Предприятие 7.7 вместе с SQL-сервером. В этом случае текст запроса преобразуется в запрос на SQL и выполняется на сервере. Результат запроса — временный набор данных — передается на клиентскую машину. Поскольку выборка обычно существенно меньше объема всей базы данных, то происходит уменьшение нагрузки на локальную сеть.

На скорость выполнения запросов влияет правильное использование обращений к базе данных. При описании запроса нужно максимально использовать встроенные возможности языка запросов и минимально сократить использование встроенного языка 1С (там, где это возможно).

Элементарные условия

В листинге 5.10 приведен текст запроса, выбирающего договоры по конкретному агенту. При этом условие задано как `Договор.Агент в ВыбАгент`. Запрос при формировании выборки сам отрабатывает только *элементарные условия*, не запуская при этом исполнительную среду встроенного языка, что значительно экономит время. Элементарные условия выглядят следующим образом:

`<ЛеваяЧастьУсловия> <ЗнакСравнения> <ПраваяЧастьУсловия>`,

где `<ЛеваяЧастьУсловия>` и `<ПраваяЧастьУсловия>` могут быть константой, переменной запроса или внешней переменной; `<ЗнакСравнения>` — логический оператор {>, <, =, <>, >=, <=, в}.

В нашем случае условие не является элементарным и, следовательно, проверяется исполнительной средой встроенного языка. Правильный текст запроса приведен в листинге 5.11.

Листинг 5.10. Неоптимальный запрос

```
Период С НачДата По КонДата;
ОбрабатыватьДокументы все;
Договор=Документ.ДоговорЗайма.ТекущийДокумент;
СуммаДоговора=Документ.ДоговорЗайма.Сумма;
Функция Сумма=Сумма(СуммаДоговора);
Группировка Договор;
Условие(Договор.Агент в ВыбАгент);
```

Листинг 5.11. Оптимальный запрос

```
Период С НачДата По КонДата;
ОбрабатыватьДокументы все;
Договор=Документ.ДоговорЗайма.ТекущийДокумент;
СуммаДоговора=Документ.ДоговорЗайма.Сумма;
Агент=Документ.ДоговорЗайма.Агент;
Функция Сумма=Сумма(СуммаДоговора);
Группировка Договор;
Условие(Агент в ВыбАгент);
```

Использование граф отбора

Использование графы отбора *(см. главу 4)* может существенно уменьшить время формирования запроса, так как задействует информацию о списке документов, накопленную системой по заданному критерию. Для того чтобы воспользоваться графами отбора, запрос должен удовлетворять следующим требованиям:

- в запросе должна быть объявлена переменная, пути которой включены в графу отбора;
- должно быть задано элементарное условие;
- условие должно быть со знаком сравнения "=".

У переменной типа `"Запрос"` существует функция `ИспользоватьГрафуОтбора (<ГрафаОтбора>)`, которая может управлять механизмом выборки данных с использованием графы отбора. По умолчанию производится автоматический выбор графы отбора.

5.3. Выполнение запроса

Запрос выполняется с помощью функции `Выполнить(<ТекстЗапроса>)`, параметром которой является строка — текст запроса. Выполнение запроса производится в три этапа. Сначала производится синтаксический контроль запроса. Если обнаруживается ошибка, то в окно сообщений выводится текст ошибки, а функция `Выполнить()` возвращает ноль. Затем создается таблица выборки (это еще не результат запроса!) и, наконец, производится накопление данных.

Если при выполнении запроса происходит ошибка в выражении, описанном на встроенном языке (например, в неэлементарном условии), то эта ошибка выдается в модальном окне, и, после нажатия кнопки **OK**, выполнение запроса продолжается.

Если при выполнении запроса оказывается, что выполнить запрос по заданным условиям не удается (например, обращение к итогам регистра после точки актуальности), то выдается сообщение об ошибке и возвращается ноль.

Если запрос успешно выполнен, то функция `Выполнить()` возвращает единицу.

Пример выполнения запроса приведен в листинге 5.12.

5.4. Обработка результатов запроса

После выполнения запроса в программе можно использовать полученный временный набор данных, который хранится в файле в формате DBF на локальном компьютере. Доступ к результату запроса производится через переменную типа `"Запрос"`. Обращаться можно к группировкам, функциям и

внутренним переменным запроса, например, `Запрос.Агент`, `Запрос.Договор`, `Запрос.Сумма`.

Изначально запрос позиционирован на первой записи временного набора данных, где содержится общий итог по запросу (если он присутствует в запросе). Поэтому общие итоги можно использовать сразу же после выполнения запроса.

Для перебора строк временного набора данных используется функция `Группировка(<Группировка>,<Направление>)`, параметром которой является либо порядковый номер, либо название группировки. Параметр `<Направление>` может принимать одно из двух значений: 1 — группировка по возрастанию, −1 — по убыванию. Функция возвращает единицу, если удалось получить следующую группировку, и ноль — в противном случае. После прохода по всем значениям группировок запрос позиционируется либо на группировке вышестоящего уровня, либо на строке итогов.

Внимание!

Нельзя обращаться к вложенной группировке, если предварительно не получено значение предыдущей группировки.

Для обработки результатов запроса обычно используется последовательность команд, приведенная в листинге 5.12.

Листинг 5.12. Последовательность команд для обработки результатов запроса

```
Запрос=СоздатьОбъект("Запрос");
ТекстЗапроса = "…";   // Создаем текст запроса на "языке запросов"
Если Запрос.Выполнить(ТекстЗапроса)<>1 Тогда
        Возврат;
КонецЕсли;
Пока Запрос.Группировка(1)=1 Цикл
        // Вывод результатов по первой группировке
        Пока Запрос.Группировка(2)=1 Цикл
        // Вывод результатов по второй группировке
        // …
        КонецЦикла;
КонецЦикла;
// Вывод результатов по всем группировкам ("Итого")
```

5.5. Конструктор запросов

Запрос можно создать с помощью конструктора запросов. Чтобы вызвать конструктор запроса, нужно открыть программный модуль, в меню

Конструкторы выбрать пункт **Запрос**. Конструктор последовательно запрашивает период запроса, внутренние переменные, функции, группировки, условия. На последнем шаге отображается текст запроса и предлагается сгенерировать процедуру, печатную форму, и добавить кнопку вызова в форму диалога.

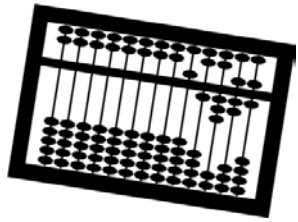
Запрос, созданный с помощью конструктора, можно повторно отредактировать с его же помощью. Если запросов в программном модуле несколько, то нужно давать им разные имена.

Задание 5.3

Создайте с помощью конструктора запрос, выводящий информацию по договорам займа с группировкой по агентам и валютам, с отбором по агенту и по заимодавцу.

5.6. Контрольные вопросы

1. Что такое запрос?
2. Какие операторы используются в запросе?
3. Какой минимальный набор операторов должен быть использован в тексте запроса, чтобы результат был ненулевым?
4. В каком порядке должны идти операторы запроса?
5. Можно ли в запросе при проверке условий использовать функции, описанные на встроенном языке 1С?
6. Можно ли в запросе в арифметических выражениях использовать функции, описанные на встроенном языке 1С?
7. Для чего применяется оператор `Без итогов`?
8. Как с помощью запроса определить общее число документов, элементов справочника?
9. Каким требованиям должен удовлетворять текст запроса, чтобы запрос выполнялся наиболее быстро?



Глава 6

Работа с таблицами

Редактор таблиц — это один из элементов технологической платформы системы 1С:Предприятие 7.7. С таблицами можно работать как в режиме конфигуратора, так и в режиме 1С:Предприятия; как вручную (вводить в ячейки таблиц информацию), так и программно (создавать и заполнять таблицу с помощью встроенного языка).

В основном таблицы используются для создания отчетов в табличном виде. Однако иногда пользователю предлагается внести какие-то данные в таблицу и затем либо сохранить их в объектах метаданных (элементах справочников, документах или бухгалтерских операциях), либо просто вывести на печать.

6.1. Редактор таблиц

Чтобы создать таблицу, запустите 1С:Предприятие 7.7 (неважно, в режиме конфигуратора или предприятия), откройте в меню **Файл** пункт **Новый**, выберите вид документа **Таблица** и нажмите кнопку **ОК**. В результате откроется форма новой таблицы. В ячейках таблицы можно вводить значения типа "Строка". С таблицей можно выполнять различные действия, которые доступны в меню **Действия**, **Таблица** и **Вид**.

Таблицу можно сохранить в файле с расширениями mxl (формат таблиц по умолчанию), xls (таблица Excel), htm (гипертекстовый документ), txt (текстовый формат). Сохраненную таблицу в формате mxl можно заново открыть с помощью меню **Файл**, пункт **Открыть**.

В конфигурации таблицы обычно используются как шаблоны отчетов и могут располагаться в формах объектов и в разделе общих таблиц (раздел общих таблиц вызывается из контекстного меню, если встать в корень дерева метаданных). При этом общие таблицы относятся к глобальному контексту, а таблицы из форм объектов — к локальному контексту.

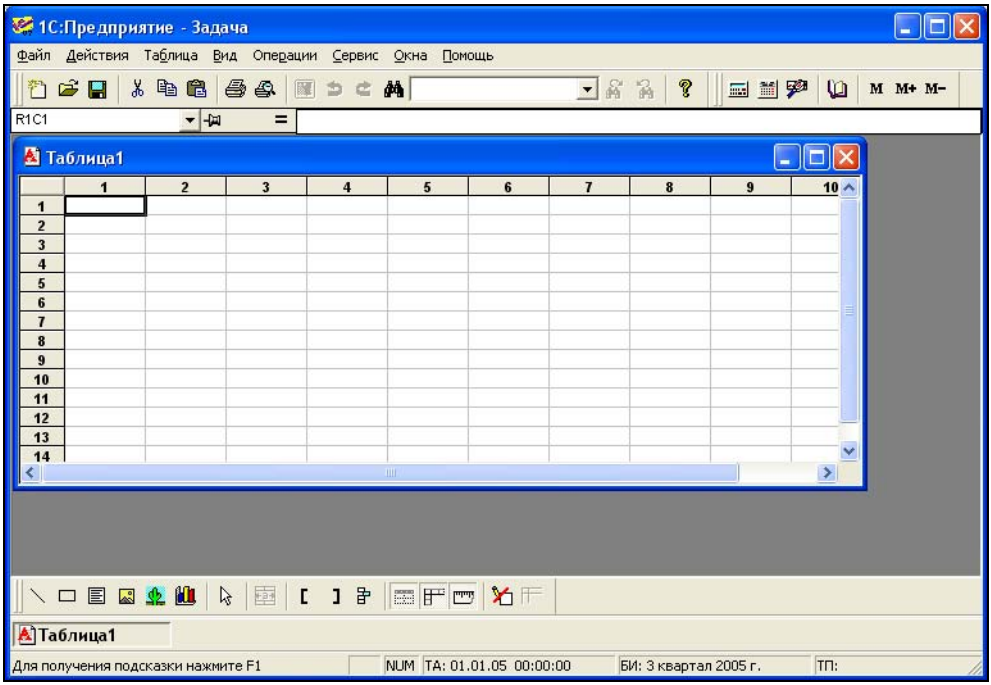


Рис. 6.1. Новая таблица

Обычно при создании формы объекта автоматически создается одна таблица. Чтобы добавить, удалить, копировать или изменить имя таблицы, нужно подвести курсор мыши к нижним вкладкам формы, нажать правую кнопку мыши и выполнить необходимое действие из контекстного меню.

6.2. Работа с таблицами на встроенном языке

Для работы с таблицами на встроенном языке используется агрегатный тип данных "Таблица", который предоставляет множество разнообразных функций. Приведем некоторые из них:

- `Открыть(<ИмяФайла>)` — открыть существующую таблицу в формате MXL;
- `Показать([<Заголовок>])` — открыть окно с таблицей для просмотра и редактирования;
- `Записать(<ИмяФайла>, <ТипФайла>)` — записать таблицу в файл с указанным именем и форматом (MXL, XLS, HTM, TXT);
- `Напечатать(<Флаг>)` — отправить таблицу на печать с открытием диалога печати (`Флаг` равен `1`) или без открытия диалога печати (`Флаг` равен `0`).

6.2.1. Режимы работы с таблицей

При работе с таблицей есть два принципиально разных режима:

- *обычный режим* — в ячейках таблицы находятся строки и невидимые поля произвольного типа, называемые "расшифровкой";
- *режим "Для ввода данных"* — в ячейках таблицы находятся значения произвольного типа, здесь также можно задавать формулы — выражения на встроенном языке, которые выполняются непосредственно после ввода данных в ячейку. Формулой, заданной в ячейке, определяется, как изменятся данные в других ячейках.

Внимание!

Отметим, что режим "Для ввода данных" доступен только для форм отчетов и обработок. В этом режиме используются другие атрибуты и методы работы с таблицей, отличные от обычного режима.

6.2.2. Работа с ячейками таблицы в обычном режиме

Для работы с ячейками таблицы используется функция `Область(<R1>, <C1>, <R2>, <C2>)`, которая возвращает значение типа "ОбластьТаблицы". Параметрами `<R1>` и `<R2>` задаются начальная и конечная строки области, `<C1>` и `<C2>` — начальная и конечная колонки области. Нумерация колонок и строк начинается с единицы. В свою очередь, у области таблицы есть различные функции форматирования (шрифт, цвет, рамки, положение и др.) и, если область состоит из одной ячейки, функции чтения и записи содержимого ячейки.

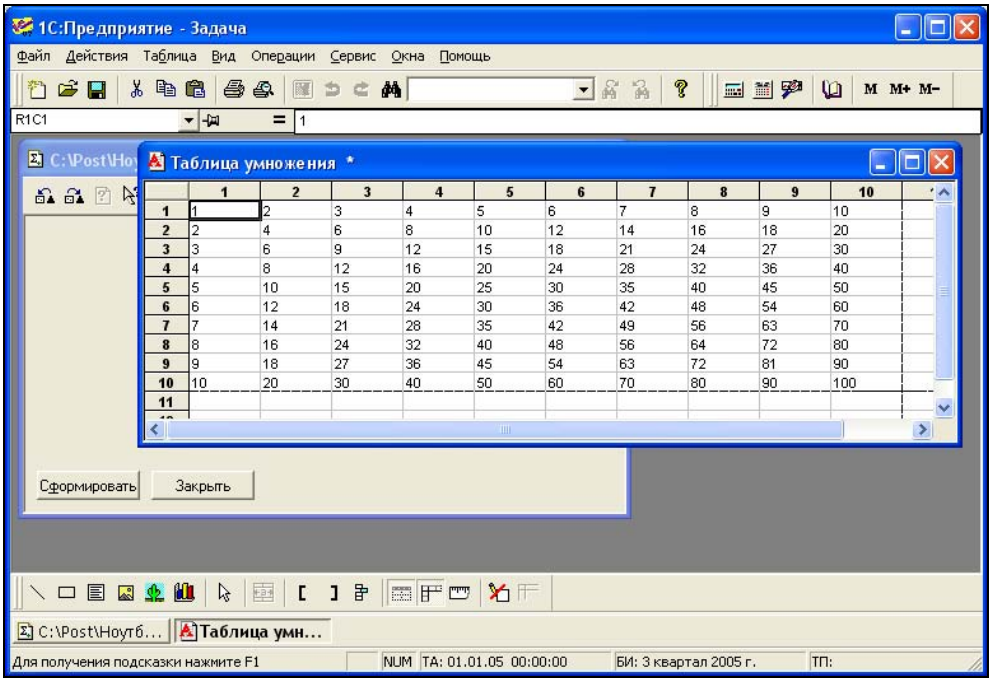
Чтение или запись *видимого* поля ячейки таблицы производится через атрибут области таблицы `Текст`. Чтение или запись *невидимого* поля ячейки таблицы осуществляется с помощью функции области таблицы `Расшифровка(<Значение>)`.

Приведем пример. Создадим таблицу и заполним ее значениями произведения номера колонки на номер строки, как показано в листинге 6.1. Полученный результат приведен на рис. 6.2.

Листинг 6.1. Таблица умножения

```
Таб=СоздатьОбъект("Таблица");
Для Стр=1 По 10 Цикл
Для Кол=1 По 10 Цикл
// Выводим текст (видимое поле ячейки)
```

```
Таб.Область(Стр,Кол).Текст=Стр*Кол;
// Выводим расшифровку (невидимое поле ячейки)
Таб.Область(Стр,Кол).Расшифровка(""+Стр+"*"+Кол+ "="+Стр*Кол);
КонецЦикла;
КонецЦикла;
Таб.Показать("Таблица умножения");
```



	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100
11										

Рис. 6.2. Таблица умножения

Как мы видим, значения вывелись корректно, но не хватает форматирования. Добавим рамки, установим выравнивание справа, как показано в листинге 6.2. Естественно, что форматирование нужно вставить до вывода окна таблицы функцией `Показать()`. Результат приведен на рис. 6.3.

Листинг 6.2. Форматирование области таблицы

```
Обл=Таб.Область(1,1,10,10);
Обл.Рамка(3,3,3,3); // 3 - тонкая сплошная линия
Обл.РамкаОбвести(5,5,5,5); // 5 - толстая сплошная линия
Обл.ГоризонтальноеПоложение(2);
```

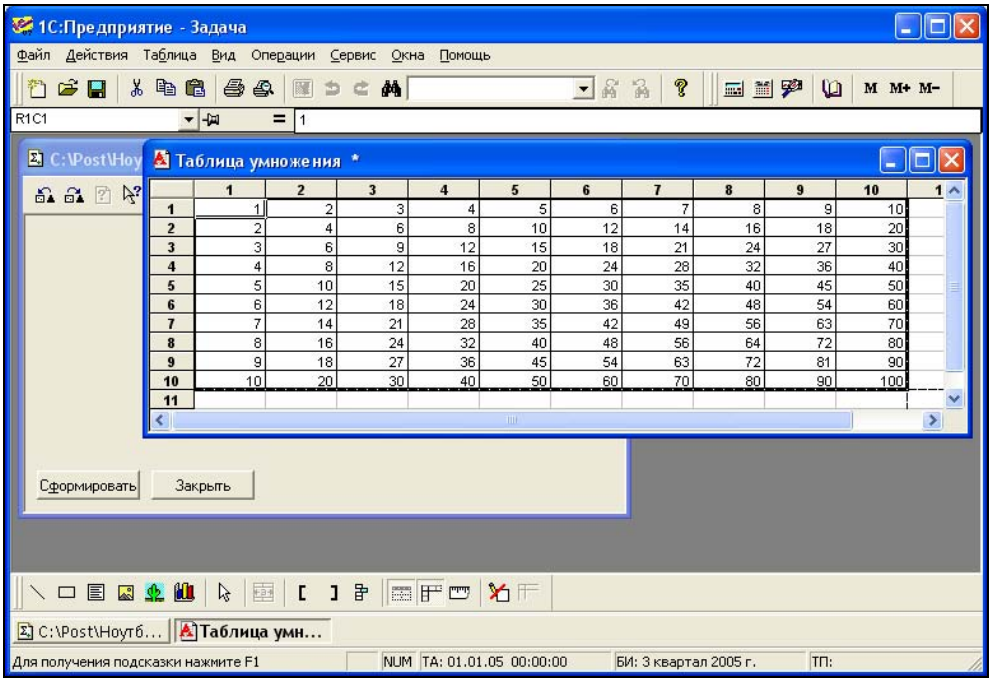


Рис. 6.3. Форматирование таблицы умножения

В таблице мы видим только текст, а как можно посмотреть расшифровки ячеек? Это можно сделать двумя способами:

- посмотреть свойства ячейки — тогда мы увидим значение расшифровки во внутреннем представлении;
- включить режим **Только просмотр** из меню **Вид** и щелкнуть два раза кнопкой мыши по ячейке таблицы — тогда запустится стандартная процедура обработки ячейки таблицы.

Для документов и справочников открывается их форма, значения базовых типов открываются в модальном окне. Можно определить собственную процедуру обработки ячейки таблицы, это будет описано в *разд. 6.2.4*.

Задание 6.1

1. Модифицируйте отчет, созданный по заданию 4.3, таким образом, чтобы информация о договорах займа выводилась в таблицу.
2. Создайте внешний отчет "`ТаблицаУмножения`" на основании листингов 6.2 и 6.3. Выполните отчет и посмотрите значения расшифровок ячеек.
3. Создайте отчет "`БиномиальныеКоэффициенты`", который выводит в таблицу биномиальные коэффициенты, вычисляемые по формуле:

$$C(n, m) = n!/(m! \times (n - m)!).$$

4. Напишите процедуру транспонирования таблицы размера *m* на *n*. Процедура транспонирования заключается в том, что строки меняются местами со столбцами. Например, следующая таблица:

   1 2 3

   4 5 6

   будет преобразована в таблицу:

   1 4

   2 5

   3 6.

6.2.3. Использование шаблонов таблицы

Рассмотренный способ заполнения таблицы отчета является универсальным, но в то же время достаточно трудоемким, так как требует все форматирование выполнять в программном модуле. Удобнее создать шаблон таблицы визуально — нарисовать рамки, задать шрифты, выравнивание и т. д.

Исходная таблица

Для этого у объекта "Таблица" есть функция ИсходнаяТаблица(<ИмяТаблицы>), которая позволяет задать таблицу-шаблон. Исходную таблицу с заданным именем ищут сначала в форме объекта, затем в общих таблицах и, наконец, во внешнем файле с расширением mxl.

Замечание

В восьмой версии 1С:Предприятия таблицы-шаблоны стали называть *макетами*.

Если таблица-шаблон не задана явно функцией ИсходнаяТаблица(), то в качестве таблицы-шаблона используется первая по порядку таблица формы объекта, в котором выполняется программный модуль.

Секция таблицы

Поскольку форматирование части таблицы обычно повторяется несколько раз (например, строки расходной накладной), то имеет смысл в таблице-шаблоне рисовать только часть, а не всю таблицу, тем более что количество строк и колонок может быть заранее неизвестно (например, число строк расходной накладной зависит от числа товаров в документе). Для обращения к части таблицы используются *секции таблицы*. Секции бывают горизонтальные (включают одну или несколько строк) или вертикальные (вклю-

чают одну или несколько колонок), вложенные (одна строка или колонка принадлежит нескольким секциям).

Чтобы создать секцию таблицы, нужно выделить строки или столбцы таблицы (навести курсор мыши на заголовки строк или столбцов и, удерживая левую кнопку мыши, выделить необходимые строки или столбцы), затем в меню **Таблица** выбрать пункт **Включить в секцию**. Далее нужно задать идентификатор секции, который будет отображаться на полях таблицы. Если какие-то строки необходимо исключить из секции, то следует выполнить аналогичные действия, только в меню **Таблица** выбрать пункт **Исключить из секции**.

Операцию по включению строк в секцию можно выполнить повторно, тогда появятся вложенные секции. На рис. 6.4 приведены примерные названия и содержание секций таблицы. Отметим, что порядок секций не играет никакой роли, и названия горизонтальных и вертикальных секций не должны совпадать.

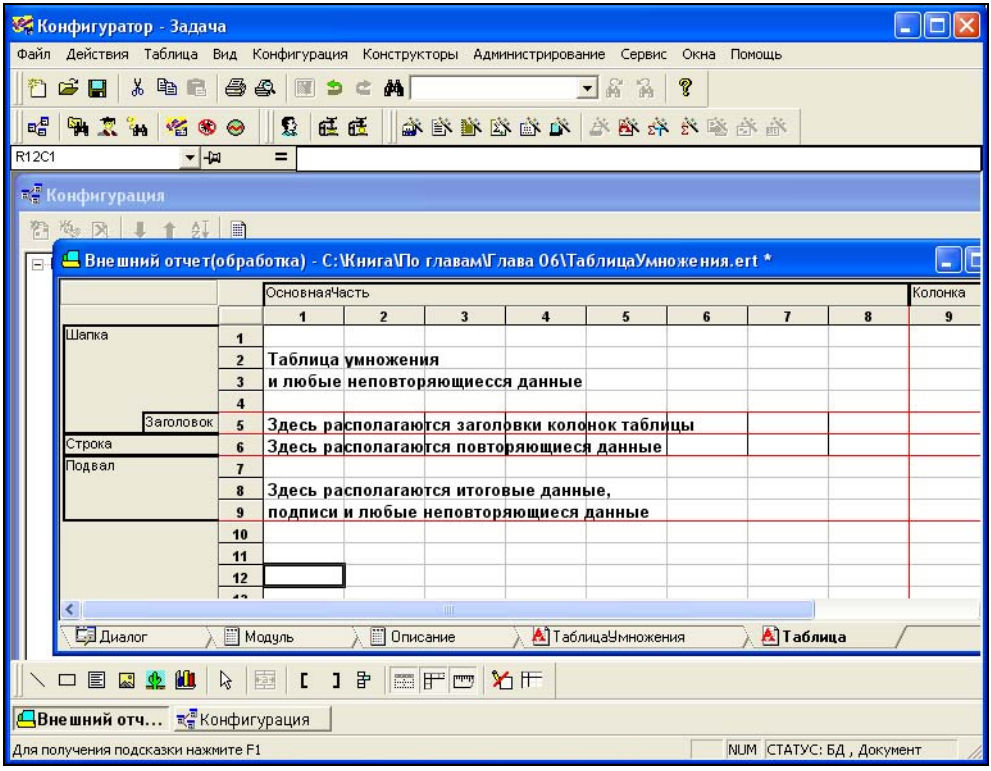


Рис. 6.4. Секции таблицы

Ячейки таблицы-шаблона

В ячейках таблицы-шаблона может располагаться текст, выражение на встроенном языке, шаблон или фиксированный шаблон. Для того чтобы задать тип ячейки, нужно выбрать ячейку (или выделить группу ячеек), нажать правую кнопку мыши и выбрать пункт **Свойства**. Откроется форма редактирования свойств ячейки (рис. 6.5).

Внимание!

Если выделить целиком всю строку и указать тип ячеек "Выражение", то на самом деле тип останется прежним, хотя в свойствах будет показываться тип "Выражение". Поэтому никогда не задавайте тип ячейки сразу для всей строки.

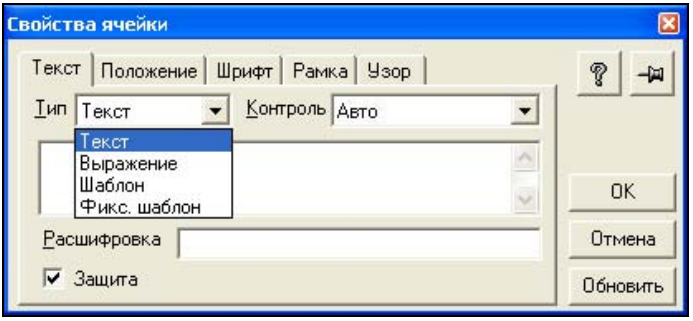


Рис. 6.5. Свойства ячейки

Тип ячейки "Шаблон" представляет собой комбинацию текста и выражения, при этом выражения записываются в квадратных скобках, например, как показано на рис. 6.6. Здесь `КоличествоСтрок` и `КоличествоСтолбцов` — это переменные, которые должны быть определены до вычисления выражения.

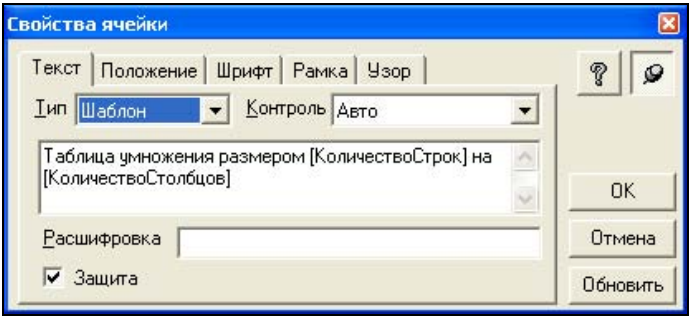


Рис. 6.6. Ячейка типа"Шаблон"

Чтобы задать значение расшифровки, нужно записать выражение в поле с соответствующим названием (рис. 6.7).

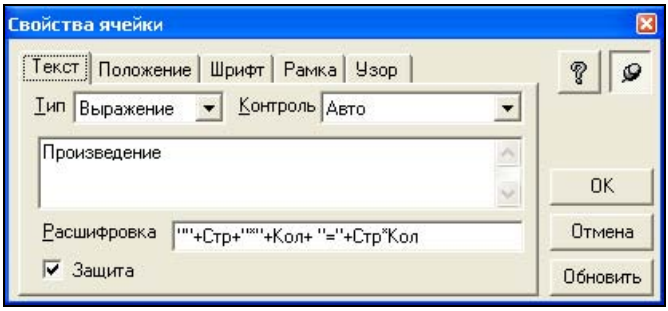


Рис. 6.7. Заполнение поля **Расшифровка**

Заполнение таблицы

Для переноса ячеек из таблицы-шаблона в текущую используются следующие функции таблицы:

- `Вывести()` — целиком переносит исходную таблицу-шаблон в текущую таблицу;
- `ВывестиСекцию(<Секция>)` — переносит секцию исходной таблицы-шаблона в текущую таблицу. Вывод начинается с новой строки;
- `ПрисоединитьСекцию(<Секция>)` — присоединяет секцию исходной таблицы-шаблона в текущую строку справа.

В качестве параметра `<Секция>` можно задавать имена горизонтальной и вертикальной секций, разделенных вертикальной чертой, например, `"Строка|Колонка"`.

Перепишем вывод таблицы умножения с использованием таблицы-шаблона. Для этого во внешний отчет `"ТаблицаУмножения"` добавим два реквизита диалога `КоличествоСтрок` и `КоличествоСтолбцов` числового типа, в которых пользователь будет задавать размер таблицы. Далее, добавим таблицу с названием `"ТаблицаУмножения"` и отформатируем ее согласно рис. 6.8.

Внимание!

При просмотре таблицы шаблоны и выражения автоматически отображаются с угловыми скобками. Самим заключать выражения и шаблоны в угловые скобки не надо!

Теперь напишем процедуру формирования отчета, как показано в листинге 6.3. Полученный результат приведен на рис. 6.9.

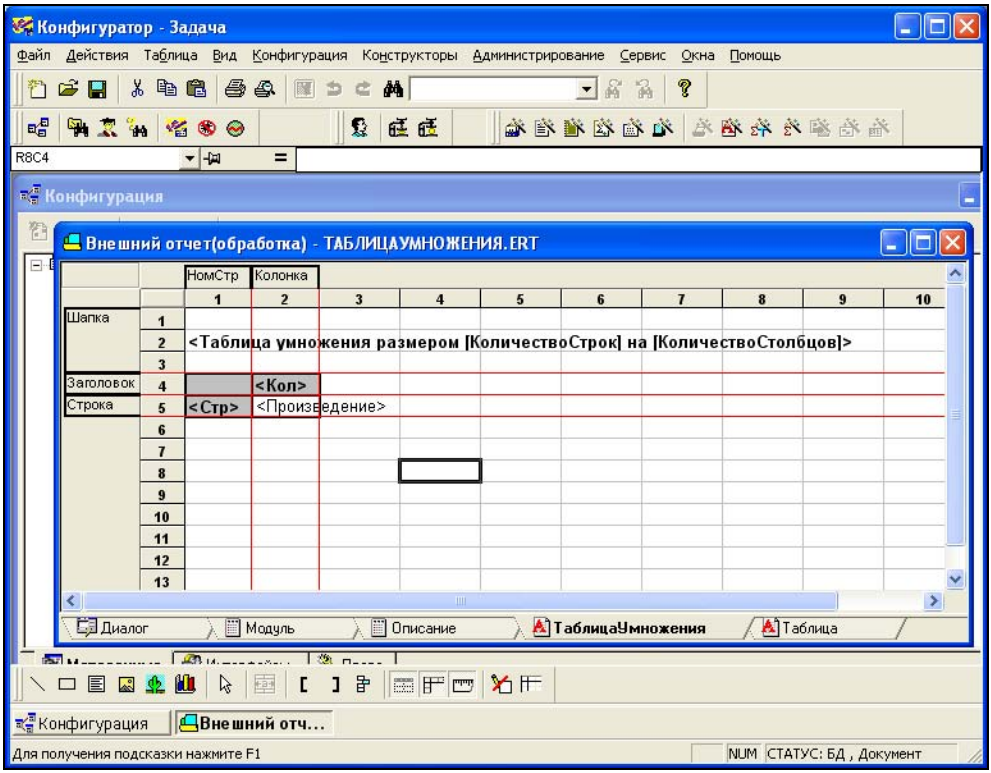
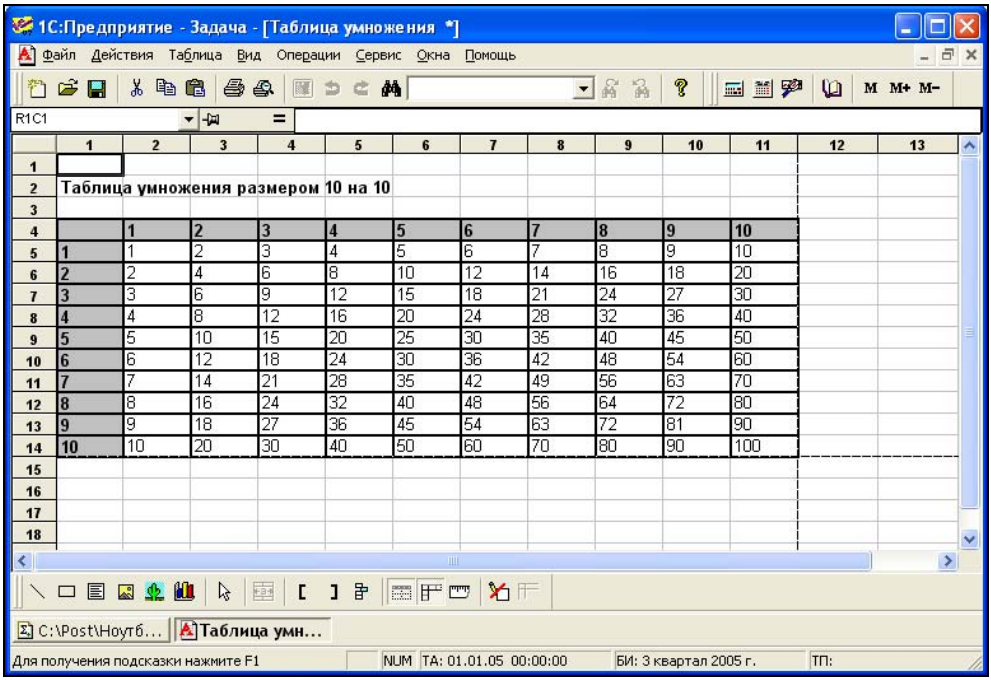


Рис. 6.8. Таблица-шаблон для отчета "ТаблицаУмножения"

Листинг 6.3. Вывод таблицы умножения с использованием шаблона

```
Таб=СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("ТаблицаУмножения");
// Выводим шапку
Таб.ВывестиСекцию("Шапка");
Таб.ВывестиСекцию("Заголовок|НомСтр");
Для Кол=1 По КоличествоСтолбцов Цикл
        Таб.ПрисоединитьСекцию("Заголовок|Колонка");
КонецЦикла;
// Выводим таблицу умножения
Для Стр=1 По КоличествоСтрок Цикл
        Таб.ВывестиСекцию("Строка|НомСтр");
        Для Кол=1 По КоличествоСтолбцов Цикл
                Произведение = Стр*Кол;
```

```
                    // Выводим текст (видимое поле ячейки)
                    Таб.ПрисоединитьСекцию("Строка|Колонка")
        КонецЦикла;
КонецЦикла;
Таб.Показать("Таблица умножения");
```



	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Рис. 6.9. Отчет "ТаблицаУмножения"

Задание 6.2

1. Переделайте отчеты из задания 6.1 с использованием таблицы-шаблона.
2. Создайте печатную форму списка агентов, которая должна вызываться из формы списка справочника агентов.
3. Создайте печатную форму списка заимодавцев, которая должна вызываться из формы списка справочника заимодавцев.
4. Создайте печатную форму документа "ДоговорЗайма", которая должна вызываться из формы документа.

Указание

Для быстрой разработки печатных форм документов и справочников удобно использовать конструктор печати.

5. Создайте отчет по договорам в виде "шахматки": по строкам таблицы расположите агентов, по колонкам — заимодавцев, а в ячейке на пересечении строки и столбца должна размещаться сумма договоров. Для извлечения данных используйте запрос.

	Сбербанк	**КМБ**	**...**	
Иванов				
Петров				
...				

После того как сформирован отчет "ТаблицаУмножения", заметим, что есть определенное неудобство в его использовании: форма, на которой мы задаем размеры таблицы, находится в одном окне, а форма таблицы — в другом. К тому же, каждый раз создается новая таблица, а не перерисовывается старая.

Для того чтобы при нажатии кнопки **Сформировать** таблица не создавалась каждый раз заново, нужно создавать таблицу при открытии формы; а чтобы она была доступна в процедуре "Сформировать ()", нужно объявить ее переменной программного модуля, как показано в листинге 6.4.

Листинг 6.4. Работа с одной таблицей

```
Перем Таб;
Процедура Сформировать()
// Предварительно очищаем таблицу
Таб.Очистить();
// Далее листинг 6.3 без первой строки
КонецПроцедуры
Таб=СоздатьОбъект("Таблица");
```

Такой способ позволяет перерисовывать одну и ту же таблицу, но все равно остается два окна: параметры нужно вводить в одной форме, а рисовать таблицу — в другой.

6.2.4. Обработка ячеек таблицы

Когда таблица открыта в режиме только просмотра и пользователь наводит курсор мыши на ячейку с непустой расшифровкой, то курсор мыши меняется на крестик с лупой. Это означает, что данную ячейку можно расшифровать (детализировать) по двойному нажатию левой кнопки мыши.

При этом срабатывает механизм стандартной обработки расшифровки ячейки, который для ссылки на документ открывает форму документа, для ссылки на элемент справочника — форму элемента справочника, а для значений типа "Строка", "Число" или "Дата" открывает модальное окно с текстом расшифровки и кнопкой **ОК**.

Чтобы задать какие-либо действия при обработке ячейки таблицы, нужно в форме программного модуля, откуда была открыта таблица, описать предопределенную процедуру `ОбработкаЯчейкиТаблицы(<Значение>, <ФлагСтандартнойОбработки>, <КонтекстТаблицы>)`.

Замечание

Если в модуле формы, из которой была вызвана таблица, не определена такая функция, то будет вызываться функция с таким же именем из глобального модуля.

Например, можно обновить таблицу более свежими данными, открыть форму настройки параметров таблицы или запустить форму более детального отчета. Примеры всех перечисленных действий можно найти в типовой конфигурации "Бухгалтерский учет".

Рассмотрим этот механизм на примере нашего отчета `"ТаблицаУмножения"`. Для этого изменим диалог, таблицу-шаблон и модуль формы следующим образом: на форме диалога оставим только кнопки **Сформировать** и **Закрыть**. Таблица-шаблон приведена на рис. 6.10. Мы добавили строку в начало таблицы с текстом и расшифровкой "Обновить", а также две строки с количеством строк и столбцов таблицы. Ячейки со значениями количества строк и столбцов мы обвели толстой сплошной линией (для наглядности) и сняли с них защиту (флажок в нижней части настройки свойств ячейки). Если с ячейки снята защита, то ввод в эту ячейку возможен даже в режиме **Только просмотр**.

Алгоритм работы отчета будет следующий: пользователь в сформированной таблице отчета будет изменять количество строк или столбцов. Далее по двойному щелчку левой кнопки мыши над ячейкой **Обновить** будет запускаться наша процедура `ОбработкаЯчейкиТаблицы()`, которая будет перерисовывать таблицу. Текст программного модуля приведен в листинге 6.5.

Листинг 6.5. Модуль отчета `"ТаблицаУмноженияВвод"`

```
Перем Таб;
Процедура Сформировать(Таб,КоличествоСтрок,КоличествоСтолбцов)
Таб.Очистить();
Таб.ИсходнаяТаблица("ТаблицаУмножения");
// Выводим шапку
Таб.ВывестиСекцию("Шапка");
```

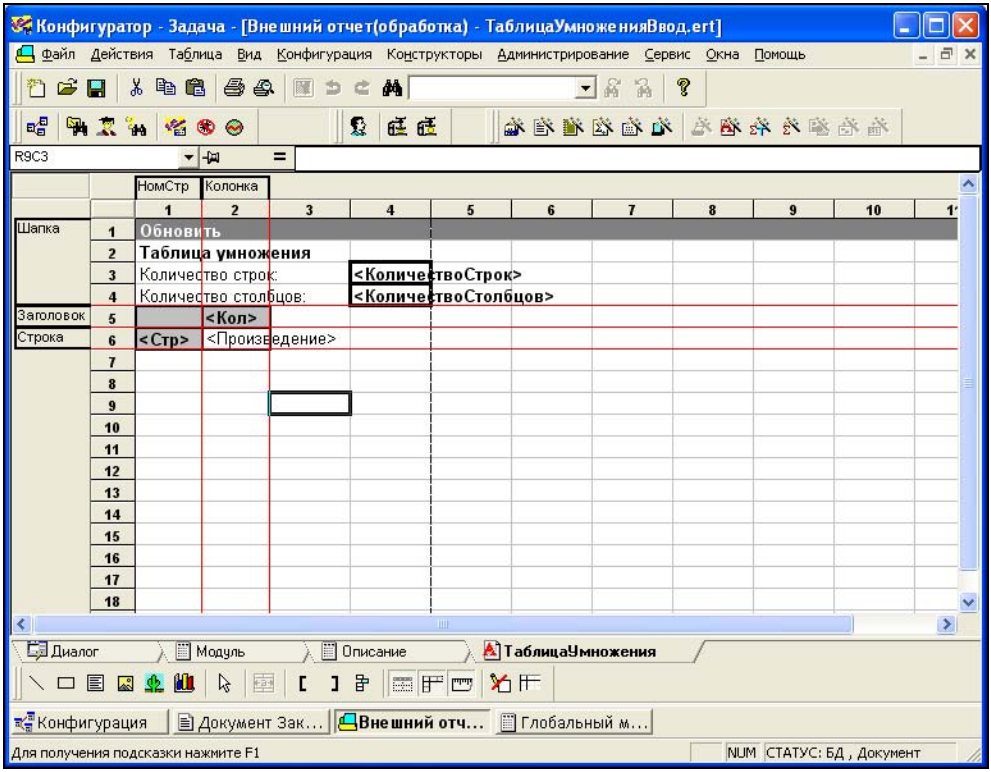


Рис. 6.10. Таблица-шаблон отчета "ТаблицаУмноженияВвод" с возможностью обновления таблицы

```
Таб.ВывестиСекцию("Заголовок|НомСтр");
Для Кол=1 По КоличествоСтолбцов Цикл
        Таб.ПрисоединитьСекцию("Заголовок|Колонка");
КонецЦикла;
// Выводим таблицу умножения
Для Стр=1 По КоличествоСтрок Цикл
        Таб.ВывестиСекцию("Строка|НомСтр");
        Для Кол=1 По КоличествоСтолбцов Цикл
                Произведение = Стр*Кол;
                // Выводим текст (видимое поле ячейки)
                Таб.ПрисоединитьСекцию("Строка|Колонка")
        КонецЦикла;
КонецЦикла;
Таб.ТолькоПросмотр(1);
Таб.Показать("Таблица умножения");
```

```
КонецПроцедуры
// Описываем предопределенную процедуру
Процедура ОбработкаЯчейкиТаблицы(Расшифровка, Флаг, Таб)
Если Расшифровка="Обновить" Тогда
        // Считываем введенное пользователем количество строк и столбцов
        КоличествоСтрок= Число(Таб.Область(3,4).Текст);
        КоличествоСтолбцов= Число(Таб.Область(4,4).Текст);
        // Перерисовываем таблицу
        Сформировать(Таб, КоличествоСтрок,КоличествоСтолбцов);
        Флаг=0; // Отключаем стандартные действия
КонецЕсли;
КонецПроцедуры
Таб=СоздатьОбъект("Таблица");
```

6.2.5. Совмещение диалога и таблицы на одной форме

Система 1С:Предприятие 7.7 позволяет совместить форму диалога и таблицу только в отчете или обработке. Для этого нужно открыть в конфигураторе отчет или обработку, зайти в меню **Действия** и выбрать пункт **Свойства формы** (см. рис. 6.11). Далее в переключателе **Использовать таблицу** выбрать пункт **Пустую**, а в переключателе **Положение** — пункт **Снизу**. Это означает, что вместе с диалогом будет использоваться таблица, расположенная под диалогом снизу. Таблица при этом создается автоматически и имеет имя `Таблица`.

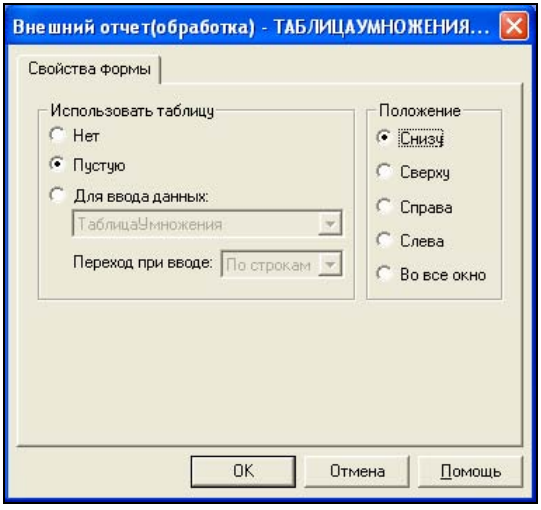


Рис. 6.11. Настройка свойств формы отчета или обработки

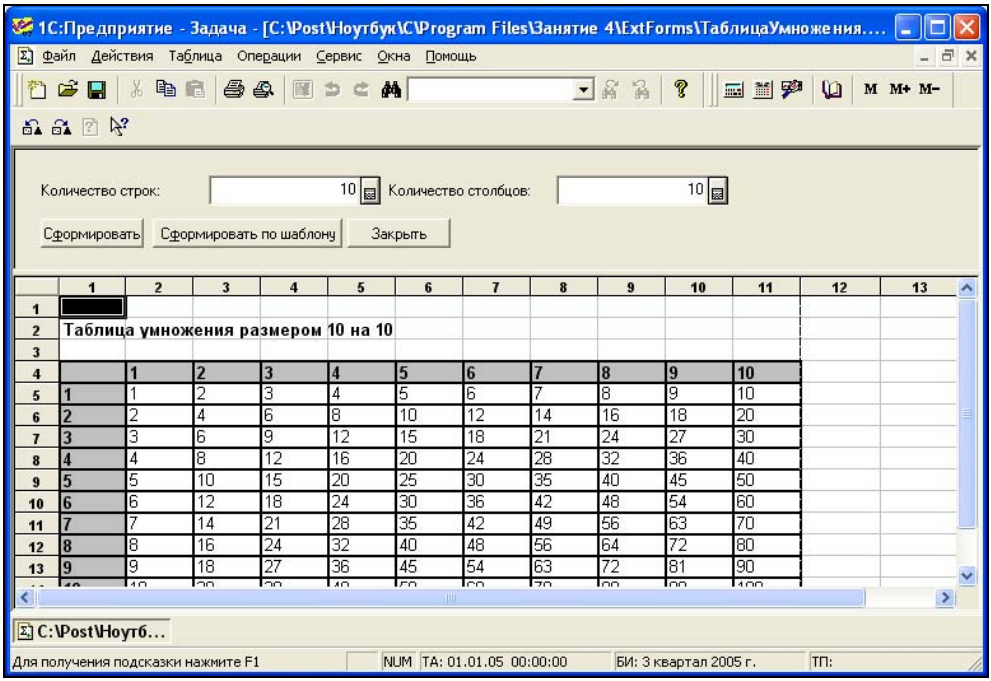
Для того чтобы модифицировать программный модуль для работы с пустой таблицей, нам нужно подправить только одну строчку, как показано в листинге 6.6.

Листинг 6.6. Работа с пустой таблицей

```
Процедура Сформировать()
Таб = Таблица; // Присваиваем ссылку на пустую таблицу
Таб.Очистить(); // Очищаем таблицу
// Далее листинг 6.3 без первой строки
КонецПроцедуры
```

Замечание

Чтобы таблица занимала больше места, нужно максимально поджать элементы диалога к верхней части формы, как показано на рис. 6.12.



	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90

Рис. 6.12. Отчет "ТаблицаУмножения" с использованием пустой таблицы

Задание 6.3

Реализуйте варианты формирования отчета по договорам займа из задания 6.2:

- с выводом в одну таблицу;

- с выводом в пустую таблицу;
- с обновлением таблицы.

Добавьте возможность изменения в таблице срочного и досрочного процентов по договору с режимом записи их в соответствующие документы `"ДоговорЗайма"`.

6.3. Работа с таблицей в режиме ввода данных

Применение таблиц в режиме ввода данных используется в специальных случаях, когда необходимо ввести большое количество данных в регламентированные формы. Например, в таком режиме производится работа в типовых конфигурациях с регламентированными отчетами "Баланс", "Отчет о прибылях и убытках" и др.

6.3.1. Режим ввода данных

Работать в режиме ввода данных могут только таблицы в отчетах и обработках. Данный режим использования таблицы включается через свойства формы (вызов свойств формы выполняется из меню **Действия** пунктом **Свойства формы**), как показано на рис. 6.13. При этом сама таблица для ввода данных должна уже существовать в этой же форме.

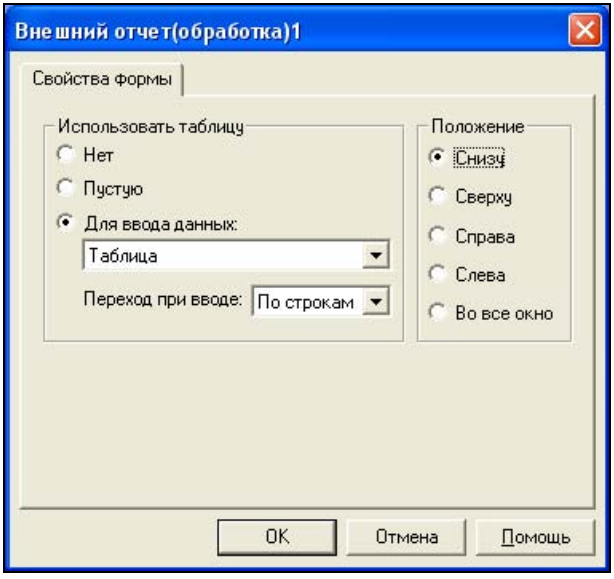


Рис. 6.13. Настройка свойств формы для ввода данных

Режим ввода данных позволяет совместить в одной форме отчета диалог формы с табличным документом или вообще заменить диалог табличным документом. У ячеек таблицы в режиме ввода в свойствах появляется дополнительная вкладка, обозначающая тип данных. Для ячеек, в которые предполагается вводить данные, требуется выключить флажок **Защита**.

Работа в режиме ввода имеет несколько особенностей с точки зрения обращения к таблице средствами встроенного языка. Во-первых, доступ к таблице осуществляется с помощью ключевого слова `Таблица`. Кроме того, в модуле формы допустимо непосредственное обращение к значениям именованных ячеек по их именам. При обращении к области таблицы используется атрибут `"Значение"`, который предоставляет доступ к значению ячеек. При этом тип значений определяется типом, выбранным на вкладке свойств ячейки **Данные**.

В свойствах ячеек можно задавать формулы. Формула выполняется непосредственно после ввода значений в соответствующую ячейку и используется обычно для вызова процедуры обработки значений других ячеек. У ячеек, имеющих признак `"Защита"`, формула является обычным выражением и вычисляет собственно значение ячейки как у элементов диалога типа "Текст".

6.3.2. Выгрузка и загрузка значений таблицы с помощью объекта *"СписокЗначений"*

Для выгрузки и загрузки большого числа переменных удобно использовать объект типа `"СписокЗначений"` *(см. главу 3)*. Для загрузки элементов из списка значений в таблицу используется функция `Загрузить(<Список>)`, причем представление списка значений становится именем ячейки, а значение записи — значением ячейки. Кроме имени можно использовать и строковое выражение вида *"RiCj"*, обозначающее ячейку в *i*-м ряду и в *j*-й колонке.

После ввода данных пользователем делается выгрузка данных из таблицы функцией `Вырузить(<Список>)`.

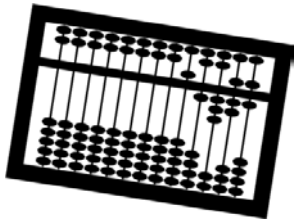
Программа, приведенная в листинге 6.7, осуществляет запись чисел от 1 до 30 в первую колонку таблицы.

Листинг 6.7. Заполнение таблицы в режиме ввода данных

```
Сп=СоздатьОбъект("СписокЗначений");
Для Н=1 по 30 Цикл
        Сп.ДобавитьЗначение(Строка(Н),"R"+Строка(Н)+"C1");
КонецЦикла;
Таблица.Загрузить(Сп);
```

6.4. Контрольные вопросы

1. Для чего используется таблица в системе 1С:Предприятие 7.7?
2. Какие режимы работы с таблицей существуют?
3. Как прочитать или записать значение из ячейки таблицы?
4. Зачем применяются таблицы-шаблоны? В чем преимущество создания отчета с помощью шаблона? В чем недостатки?
5. Как работает механизм обновления таблиц?
6. Как совместить в одной форме реквизиты диалога и таблицу?
7. Для чего используются таблицы в режиме ввода данных?



Глава 7

Работа с объектами компоненты "Бухгалтерский учет"

К объектам компоненты "Бухгалтерский учет" относятся:

- планы счетов и бухгалтерские счета;
- виды субконто;
- операции и проводки;
- бухгалтерские итоги.

Наличие бухгалтерской компоненты необходимо для того, чтобы при проведении документов формировались проводки.

7.1. Работа с бухгалтерскими счетами

Счет — это агрегатный тип данных для доступа к объектам данных — бухгалтерским счетам. Бухгалтерские счета используются для идентификации разрезов синтетического учета наличия и движения средств.

В конфигурации может быть несколько планов счетов. Объект типа "Счет" может относиться к конкретному плану счетов или быть неопределенного вида, то есть принимать значения различных планов счетов.

Структура данных объектов типа "Счет" задается в конфигураторе в разделе **Планы счетов** и является одинаковой для всех планов счетов (рис. 7.1).

У счета определены следующие атрибуты:

- `Код` — в общем случае представляет собой символьную строку вида `<Код счета>.<Код субсчета>.<Код субсчета>` и т. д.;
- `Наименование` — наименование счета;
- `Валютный` — признак ведения валютного учета;

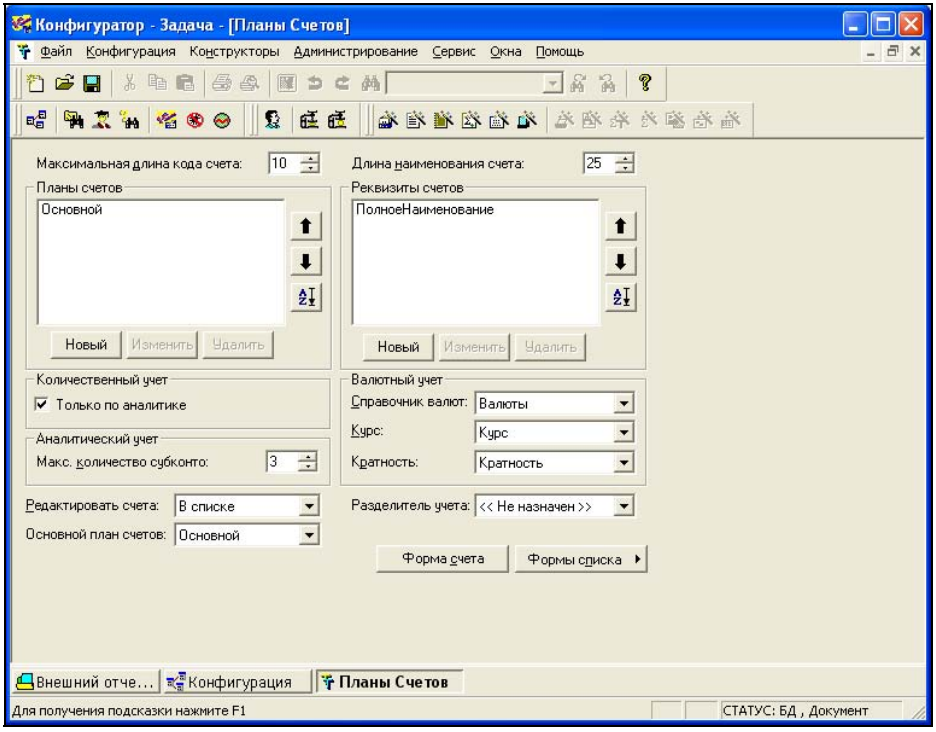


Рис. 7.1. Настройка планов счетов

- `Количественный` — признак ведения количественного учета;
- `Забалансовый` — признак того, что счет является забалансовым;
- `Активный` — принимает три значения: 1 — активный, 2 — пассивный, 3 — активно-пассивный.

Наиболее важные функции:

- `КоличествоСубконто()` — возвращает количество субконто у текущего счета;
- `ВидСубконто()` — устанавливает/возвращает вид субконто по порядковому номеру.

Бухгалтерскому счету соответствует агрегатный тип данных "Счет". В примере, представленном в листинге 7.1, в окно сообщений выводится список всех счетов из основного плана счетов.

Листинг 7.1. Вывод списка счетов в окно сообщений

```
Сч=СоздатьОбъект("Счет");
Сч.ИспользоватьПланСчетов(ПланыСчетов.Основной)
```

```
Сч.ВыбратьСчета();
Пока Сч.ПолучитьСчет()=1 Цикл
        Сообщить(Сч.Код + " " + Сч.Наименование);
КонецЦикла;
```

В примере, приведенном в листинге 7.2, создается новый счет. В конфигурации предварительно должны быть определены виды субконто: "Номенклатура", "МестаХранения".

Листинг 7.2. Создание нового счета

```
Сч=СоздатьОбъект("Счет");
Сч.ИспользоватьПланСчетов(ПланыСчетов.Основной)
Сч.Новый(0); // Счет не имеет субсчетов
Сч.Код = "41.1";
Сч.Наименование = "Товары на складах";
Сч.ВидСубконто(1,ВидыСубконто.Номенклатура);
Сч.ВидСубконто(2,ВидыСубконто.МестаХранения);
Сч.Количественный=1;
Сч.Валютный =0;
Сч.Забалансовый=0;
Сч.Активный=1;
Сч.Записать();
```

Системная функция `СчетПоКоду(<КодСчета>, [<ПланСчетов>])` возвращает ссылку на счет из соответствующего плана счетов.

Для организации аналитического учета служит объект метаданных `ВидыСубконто`. Добавление новых видов субконто осуществляется вручную в конфигураторе. `"ВидСубконто"` может принимать значения базовых типов, ссылок на объекты и перечисления. Причем для одного и того же объекта (например, справочник "Контрагенты") может существовать несколько видов субконто (например, "Комитенты" и "Комиссионеры"). В этом случае на одном счете мы можем вести учет в двух измерениях по одному и тому же справочнику. В то же время на счете не может быть двух одинаковых видов субконто.

Задание 7.1

В конфигурации, разработанной в *главе 4*:

1. Создайте план счетов "Основной" с максимальным количеством субконто — 3.

2. Создайте три вида субконто:
   - "Агенты" (тип `Справочник.Агенты`);
   - "Заимодавцы" (тип `Справочник.Заимодавцы`);
   - "ДоговорыЗайма" (тип `Документ.ДоговорЗайма`).
3. Введите следующие счета в плане счетов "Основной" вручную в режиме конфигуратора:
   - "51" — расчетный счет в рублях;
   - "52" — расчетный счет в валюте;
   - "66" — расчеты по займам, имеет три субконто: "Агенты", "Заимодавцы", "ДоговорыЗайма", а также следующие субсчета:
     - "66.3" — расчеты по займам в рублях;
     - "66.33" — расчеты по займам в валюте;
     - "66.4" — проценты в рублях;
     - "66.44" — проценты в валюте;
   - "91.2" — прочие расходы.

Внимание!

Обратите внимание на то, что произошло со списком счетов после добавления счета 91.2.

7.2. Работа с операциями и проводками

Для отражения в бухгалтерском учете информации о движении средств используется объект `"Операция"`, являющийся контейнером для хранения бухгалтерских проводок.

Объект типа `"Операция"` используется для формирования и анализа проводок, формируемых документом. Для этого у агрегатного объекта `"Документ"` существует атрибут `"Операция"`, который обеспечивает доступ к операции данного документа. Чтобы документ формировал операцию, необходимо, чтобы в документе стоял флажок **Бухгалтерский учет**.

Объект `"Операция"` также доступен непосредственно в контекстах форм "Операции", "Журнал операций" и "Журнал проводок".

Объект `"Операция"` используется для перебора существующих операций и проводок при формировании отчетов и других выборок. В этом случае объект создается с помощью вызова функции `СоздатьОбъект("Операция")`.

7.2.1. Реквизиты операции

Операция имеет следующие реквизиты:

- Шапка: `"ДатаОперации"`, `"Содержание"`, `"СуммаОперации"`, `"Документ"` (документ, которому принадлежит операция);
- Табличная часть (проводка): `"Сумма"` — сумма текущей проводки операции; `"Валюта"` — валюта текущей проводки; `"ВалСумма"` — валютная сумма текущей проводки; `"Количество"` — количество текущей проводки; `"Дебет"` — обращение к дебету проводки, `"Кредит"` — обращение к кредиту проводки.

Объекты `"Дебет"` и `"Кредит"` имеют атрибут `"Счет"` и функцию установки/получения значения субконто — `Субконто()` по его порядковому номеру при определении счета в плане счетов.

7.2.2. Обработка операций и проводок

Чтобы перебрать все операции за заданный период, необходимо открыть выборку операций функцией `ВыбратьОперации(Дата1,Дата2)` и перебрать все операции в цикле функцией `ПолучитьОперацию()`, как показано в листинге 7.3.

Листинг 7.3. Просмотр операций за заданный период

```
Опер=СоздатьОбъект("Операция");
Опер.ВыбратьОперации(Дата1,Дата2);
Пока Опер.ПолучитьОперацию()=1 Цикл
Сообщить("Операция"+Опер.Документ.НомерДок+"от"+Опер.ДатаОперации);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку()=1 Цикл
        Сообщить("Д"+Опер.Дебет.Счет.Код + "," +
                Опер.Дебет.Субконто(1)+ "," +
                Опер.Дебет.Субконто(2) + "," +
                Опер.Дебет.Субконто(3) + "," +
                "К"+Опер.Кредит.Счет.Код+ "," +
                Опер.Кредит.Субконто(1) + "," +
                ОперКредит.Субконто(2) + "," +
                Опер.Кредит.Субконто(3) + "," +
                Опер.Сумма);
КонецЦикла;
КонецЦикла;
```

Такая работа с операциями имеет один недостаток: чтобы получить информацию по одному или нескольким счетам, нужно перебирать все проводки. Чтобы выбрать проводки по конкретному счету (счетам), можно использовать функцию `ВыбратьОперацииСПроводками()`, которая имеет два варианта синтаксиса:

- `ВыбратьОперацииСПроводками(<НачалоПериода>, <КонецПериода>, <Фильтр>, <Валюта>, <ПланСчетов>, <РазделительУчета>)`
- `ВыбратьОперацииСПроводками(<НачалоПериода>, <КонецПериода>, <Счет>, <КорСчет>,<Флаг>, <Валюта>, <ПланСчетов>, <РазделительУчета>)`

В параметре `<Фильтр>` задаются критерии отбора проводок для включения в выборку. Если параметр не заполнен, в выборку включаются все проводки. В общем случае в параметре `<Фильтр>` могут находиться одна или несколько корреспонденций счетов или символьных строк, разделяемых точкой с запятой (`;`). Символьные строки представляют собой наборы символов, заключенные в кавычки.

Замечание

При передаче строки в явном виде в параметре внутри строки двойные кавычки задаются двумя символами двойных кавычек.

Корреспонденции имеют вид:

- *n* — проводки со счетом *n*;
- *n*, *m* — проводки в дебет счета *n* с кредита счета *m*.

Замечание

Здесь в качестве *n* и *m* может указываться звездочка (*), она обозначает любой счет. Например, *, 51 — все проводки с кредита 51 счета.

В выборку включаются все проводки, соответствующие следующим условиям:

- если в параметре `<Фильтр>` указаны корреспонденции счетов, то проводка должна соответствовать одной из этих корреспонденций;
- если в параметре `<Фильтр>` указаны строки символов, то в проводке должна содержаться хотя бы одна из этих строк: либо в содержании операции, либо в представлениях значений субконто и реквизитов проводки и операции.

Примеры представлены в табл. 7.1.

Таблица 7.1. *Примеры фильтров*

Фильтр	Отбираемые данные
"50"	Все проводки со счетом 50
"50, *"	Все проводки в дебет 50 счета

Таблица 7.1 *(окончание)*

Фильтр	Отбираемые данные
"* , 51"	Все проводки с кредита 51 счета
"50, 51"	Все проводки в дебет 50 счета с кредита 51
"51; 52"	Все проводки со счетом 51 или счетом 52

Второй вариант синтаксиса отличается тем, что в нем можно указать ссылки на счета (дебет и кредит), по которым будет формироваться выборка.

После получения выборки функцией `ВыбратьОперацииСПроводками()` следующую проводку можно получить функцией `ПолучитьПроводку()`.

Задание 7.2

Напишите отчет по проводкам аналогично листингу 7.3, но с применением функции `ВыбратьОперацииСПроводками()`. В форме диалога должен задаваться период и фильтр по счетам. Вывод должен осуществляться в таблицу.

7.2.3. Создание операции

Для создания "ручной" операции используется функция `Новая()`, после чего заполняются реквизиты операции. Чтобы добавить в операцию проводку, используется функция `НоваяПроводка()`, после чего заполняются реквизиты проводки. Чтобы операция добавилась в информационную базу, необходимо выполнить запись операции функцией `Записать()`. Например, чтобы ввести программно операцию по поступлению денежных средств на расчетный счет по договору займа, можно написать процедуру, приведенную в листинге 7.4. Отметим, что `Агент`, `Заимодавец` и `ДоговорЗайма` — это ссылки на элементы справочников "Агенты", "Заимодавцы" и документ "ДоговорЗайма" соответственно.

Листинг 7.4. Создание операции по поступлению денежных средств от заимодавца

```
Опер=СоздатьОбъект("Операция");
Опер.Новая();
Опер.ДатаОперации='05.04.2005';
Опер.НоваяПроводка();
Опер.Дебет.Счет=СчетПоКоду("51",ПланыСчетов.Основной);
Опер.Кредит.Счет=СчетПоКоду("66.3",ПланыСчетов.Основной);
```

```
Опер.Кредит.Субконто(1,Агент);
Опер.Кредит.Субконто(2,Заимодавец);
Опер.Кредит.Субконто(3,ДоговорЗайма);
Опер.Сумма=10000;
Опер.СодержаниеПроводки = "Получен займ";
Опер.Записать();
```

Замечание

Присваивать значения субконто в проводке можно двумя способами: `Опер.Кредит.Субконто(1,Агент)` и `Опер.Кредит.Агенты = Агент`.

7.3. Связь между документами и операциями

У документов есть системный реквизит `"Операция"` — ссылка на соответствующую операцию документа. А у этой операции в свою очередь есть системный реквизит `"Документ"` — ссылка на документ, который создал данную операцию. Таким образом, между документом, относящимся к компоненте "Бухгалтерский учет", и операцией есть однозначное соответствие.

Для операций, введенных вручную, есть специальный документ, который называется "Операция". Он создается автоматически в конфигурации при создании плана счетов. Таким образом, чтобы узнать, является операция "ручной" или "автоматической", нужно проверить условие, приведенное в листинге 7.5.

Листинг 7.5. Проверка способа ввода операции

```
Если Операция.Документ.Вид()= "Операция" Тогда
        // Ручная операция
Иначе
        // Введенная документом
КонецЕсли;
```

Номер операции — это на самом деле номер документа, который соответствует операции: `Операция.Документ.НомерДок`.

Формирование проводок операции производится в модуле документа в предопределенной процедуре `ОбработкаПроведения()`. В этой процедуре нужно написать алгоритм проведения. Например, для формирования проводки по закрытию договора документом `"ЗакрытиеДоговора"`, имеющего реквизит

"Договор" — ссылку на документ "ДоговорЗайма", можно написать алгоритм, приведенный в листинге 7.6.

Листинг 7.6. Модуль проведения документа "ДоговорЗайма"

```
Процедура ОбработкаПроведения()
// Определяем срок договора, процент
Срок=ДатаДок-Договор.ДатаНачала;
Если ДатаДок<Договор.ДатаОкончания Тогда
        Процент=Договор.ПроцентДосрочный;
Иначе
        Процент=Договор.ПроцентСрочный;
КонецЕсли;
// Определяем сумму процентов в валюте договора
СуммаПроцентов=Договор.Сумма*Процент*Срок/365/100;
// Создаем проводку
Операция.НоваяПроводка();
Операция.Дебет.Счет=СчетПоКоду("91.2");
Если Договор.Валюта=Константа.БазоваяВалюта Тогда
        // Договор в рублях
        Операция.Кредит.Счет = СчетПоКоду("66.4");
        Операция.Сумма=СуммаПроцентов;
Иначе
        // Договор в валюте
        Операция.Кредит.Счет = СчетПоКоду("66.44");
        Операция.ВалСумма=СуммаПроцентов;
        Операция.Валюта=Договор.Валюта;
        Операция.Сумма=СуммаПроцентов*
                Договор.Валюта.Курс.Получить(ДатаДок)/
                Договор.Валюта.Кратность.Получить(ДатаДок);
КонецЕсли;
Операция.Кредит.Агенты=Договор.Агент;
Операция.Кредит.Заимодавцы=Договор.Заимодавец;
Операция.Кредит.ДоговорыЗайма=Договор;
Операция.Записать();
КонецПроцедуры
```

Замечание

Важное отличие при формировании проводок из модуля документа (листинг 7.6) от формирования "ручной" операции в том, что объект "Операция" не создается функциями СоздатьОбъект() и Новая(), поскольку система делает это автоматически, если в настройках данного вида документов стоит признак **Создавать операцию всегда** или **Создавать операцию при проведении**.

Задание 7.3

1. Создайте документ "ЗакрытиеДоговора", содержащий единственный реквизит "Договор" (тип Документ.ДоговорЗайма). Определите для него форму документа.
2. Напишите процедуру ОбработкаПроведения() в модуле документа "ЗакрытиеДоговора" на основании листинга 7.6, но со следующими отличиями:
   - *дата начала договора* должна определяться датой проводки по поступлению денежных средств на расчетный счет от заимодавца по договору займа;
   - *сумма договора* должна определяться суммой проводки по поступлению денежных средств на расчетный счет от заимодавца по договору займа;
   - *валюта договора* должна определяться валютой проводки по поступлению денежных средств на расчетный счет от заимодавца по договору займа.
3. Создайте отчет следующего вида:

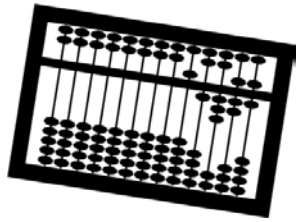
Сведения о договорах займа, завершенных до истечения установленного срока в период с . . . по . . .

Заимодавец	Агент	Договор	Дата начала договора	Срок займа по договору (в днях)	Фактическая дата завершения договора	Фактический срок займа (в днях)	Фактическая сумма займа	Начислен процент	Валюта займа
Сбербанк РФ	Иванов И. И.	ДЗ-70 от 01.04.05	05.04.05	184	01.07.2005	87	10 000	429.04	Руб.

Период, за который формируется отчет, может быть произвольным и задается в диалоге пользователем. Данные отчета должны опираться на бухгалтерские проводки.

7.4. Контрольные вопросы

1. Как организуется иерархия счетов? Чем отличаются ярлычки групп счетов при просмотре плана счетов?
2. Как определить, что операция введена вручную?
3. Играет ли роль порядок заполнения реквизитов проводки?
4. Как задать номер операции?
5. Как перебрать все проводки вида Д41.1 К60.1?
6. Какие признаки есть у субконто и что они означают?



Глава 8

Работа с бухгалтерскими итогами

В предыдущих главах мы уже познакомились с механизмами работы с операциями и проводками с помощью перебора проводок функциями переменной агрегатного типа "Операция". Однако этот механизм обладает одним существенным недостатком.

Основная информация, которая интересует бухгалтера, — это состояние бухгалтерских счетов на некоторый момент времени — *сальдо*, а также движения средств по бухгалтерским счетам — *обороты*. Для получения сальдо рассмотренными механизмами нам придется перебрать все проводки по выбранному счету с начального момента ввода данных в систему. С ростом объема введенной информации получение этих итогов будет замедляться. Выходом из этой ситуации является хранение промежуточных итогов в специальных служебных таблицах, что и сделано в системе 1С:Предприятие 7.7 в компоненте "Бухгалтерский учет". Бухгалтерские итоги хранятся в файлах 1sbkttl.dbf и 1sbkttlc.dbf.

8.1. Что хранится в бухгалтерских итогах

В бухгалтерских итогах хранятся:

- остатки и обороты по счетам с детализацией по субконто;
- обороты между счетами без детализации по субконто.

Хранение итогов поддерживается системой с детализацией до месяца.

8.2. Период бухгалтерских итогов

В пункте **Управление бухгалтерскими итогами** меню **Операции** в режиме "1С:Предприятие" устанавливается последний рассчитанный период (рис. 8.1). После последнего рассчитанного периода получить бухгалтерские итоги

невозможно. Средствами встроенного языка можно получить эту границу с помощью системной функции `КонецРассчитанногоПериодаБИ()`, которая возвращает значение типа "Дата".

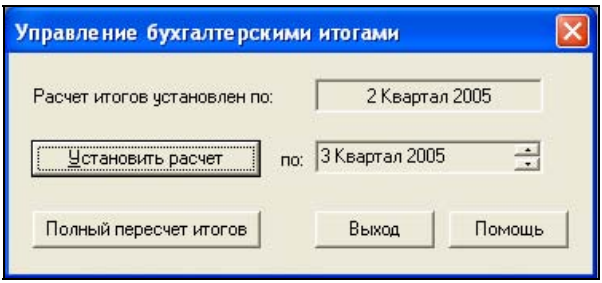


Рис. 8.1. Установка границы расчета бухгалтерских итогов

8.3. Пересчет итогов

Система 1С:Предприятие 7.7 обеспечивает автоматический пересчет бухгалтерских итогов при вводе или изменении проводок бухгалтерских операций с датой, меньшей либо равной границе рассчитанного периода бухгалтерских итогов. Если возникли подозрения, что бухгалтерские итоги некорректны (например, в результате системного сбоя), их можно пересчитать двумя способами:

- нажать кнопку **Полный пересчет итогов** в форме управления бухгалтерскими итогами (рис. 8.1);
- выполнить тестирование и исправление информационной базы *(см. главу 1)*.

8.4. Работа с бухгалтерскими итогами на встроенном языке

Обращение к бухгалтерским итогам выполняется с помощью агрегатного объекта типа "`БухгалтерскиеИтоги`". Объект может работать в трех режимах:

- с основными итогами;
- с временными итогами;
- в режиме запроса.

Функции `ИспользоватьПланСчетов()` и `ИспользоватьРазделительУчета()` позволяют назначить план счетов и разделитель учета, по которым будут выдаваться итоги.

8.4.1. Работа с основными итогами

Объект типа "БухгалтерскиеИтоги" при создании функцией СоздатьОбъект() работает в режиме основных итогов.

Внимание!

По умолчанию используется период, выбранный пользователем интерактивно, через меню **Сервис**, пункт **Параметры**, вкладка **Бухгалтерские итоги**.

Функции работы с итогами:

- ПериодМ(<Дата>) — установка периода итогов — месяц, которому принадлежит <Дата>;
- ПериодД(<ДатаНач>,<ДатаКон>) — установка периода итогов в период с <ДатаНач> по <ДатаКон>;
- СНД(<Счет>,<ТипСуммы>,<Валюта>,<Субконто1>,…) — сальдо начальное дебетовое;
- СНК(Счет>,<ТипСуммы>,<Валюта>,<Субконто1>,…) — сальдо начальное кредитовое;
- СКД(Счет>,<ТипСуммы>,<Валюта>,<Субконто1>,…) — сальдо конечное дебетовое;
- СКК(Счет>,<ТипСуммы>,<Валюта>,<Субконто1>,…) — сальдо конечное кредитовое;
- ДО(Счет>,<ТипСуммы>,<Валюта>,<Субконто1>,…) — дебетовый оборот по счету за период;
- КО(Счет>,<ТипСуммы>,<Валюта>,<Субконто1>,…) — кредитовый оборот по счету за период.

  Параметры: <Счет> — счет, <ТипСуммы> принимает значения: 1 — сумма, 2 — валютная сумма, 3 — количество; <Валюта> — значение типа "Справочник.Валюты"; <Субконто1> — значение первого субконто счета; <Субконто2> — значение второго субконто счета и т. д.;
- ОБ(<СчетДеб>,<СчетКред>,<ТипСуммы>,<Валюта>) — обороты между счетами.

Например, чтобы получить остаток товара на складе на конец текущего месяца, можно воспользоваться листингом 8.1. Переменные Товар и Склад должны ссылаться на элементы справочников "Номенклатура" и "МестаХранения" соответственно.

Листинг 8.1. Получение остатка товара на складе на конец текущего месяца

```
БИ.ПериодМ(ТекущаяДата());
Сообщить("Остаток товара " + Товар + " на складе " + Склад + " равен " +
БИ.СКД("41.1", "К",,Товар,Склад));
```

8.4.2. Работа с временными итогами

Чтобы получить итоги на любую дату, меньшую границы последнего рассчитанного периода бухгалтерских итогов (листинг 8.2), нужно выполнить временный расчет (временный, потому что он хранится только во время существования переменной типа "БухгалтерскиеИтоги") с помощью функции

Рассчитать(<ДатаНач>,<ДатаКон>,<ФильтрПоСчетам>,<ТолькоСинтетика>,<ПланСчетов>,<РазделительУчета>).

В параметре <ФильтрПоСчетам> можно задать в виде строки список счетов, разделенных запятой или точкой с запятой, по которым будет делаться расчет. Если параметр <ТолькоСинтетика> = 1, то расчет будет делаться только по счетам, иначе — по счетам и субконто.

Листинг 8.2. Получение остатка товара на складе на текущую дату

```
Если ТекущаяДата()<КонецРассчитанногоПериодаБИ() Тогда
        БИ.Рассчитать(,ТекущаяДата(), "41.1");
        Сообщить("Остаток товара " + Товар + " на складе " + Склад + "
        равен " + БИ.СКД("41.1", "К",,Товар,Склад));
КонецЕсли;
```

8.4.3. Работа в режиме запроса

Для получения более детальных бухгалтерских итогов используется режим бухгалтерского запроса. В бухгалтерском запросе выделяются группа счетов *(счета)* и группа корреспондирующих им счетов *(корсчета)*. Счета и корсчета могут детализироваться по *субконто* и корреспондирующим субконто *(корсубконто)* соответственно.

Выполнение запроса

Перед выполнением запроса устанавливаются фильтры по субконто и корреспондирующим субконто.

- ИспользоватьСубконто(<ВидСубконто>,<Значение>,<ТипФильтра>,<ПоГруппам>) — устанавливать режим отбора итогов в разрезе субконто. Параметр <ВидСубконто> задается выражением типа ВидСубконто или строкой, содержащей имя идентификатора вида субконто. Параметр <Значение> задает конкретное значение субконто. Если <ТипФильтра>=1, тогда итоги будут разворачиваться по этому виду субконто. Если <ТипФильтра>=2, тогда итоги будут отбираться по значению субконто. Если <ТипФильтра>=3, тогда это субконто вообще не будет учитываться.

  Функцию ИспользоватьСубконто() можно выполнять несколько раз для задания в запросе нескольких видов субконто. Обращение к субконто

производится по порядковому номеру (порядок определяется последовательностью команд `ИспользоватьСубконто()`).

- `ИспользоватьКорСубконто(<ВидСубконто>,<Значение>,<ТипФильтра>, <ПоГруппам>)` — устанавливать режим отбора итогов по корреспондирующим счетам в разрезе субконто.

Затем выполняется сам запрос функцией

```
ВыполнитьЗапрос(<ДатаНач>,<ДатаКон>,<ФильтрПоСчетам>,
<ФильтрПоКорСчетам>,<Валюта>,<ТипИтогов>,<Периодичность>, <ТипСуммы>),
```

которая возвращает `1`, если запрос выполнился успешно.

Параметр `<ТипИтогов>` принимает следующие значения: `1` — остатки и обороты по счетам, `2` — обороты между счетами, `3` — и то и другое.

Параметр `<Периодичность>` может принимать следующие значения:

- `"Период"` — промежуточные итоги не рассчитываются;
- `"Операция"` — промежуточные итоги рассчитываются по операциям;
- `"Проводка"` — по проводкам;
- `"День"` — по дням;
- `"Неделя"` — по неделям;
- `"Декада"` — по декадам;
- `"Месяц"` — по месяцам;
- `"Квартал"` — по кварталам;
- `"Год"` — по годам.

Обращение к результатам запроса

Для перебора группировок используются следующие функции:

- `ВыбратьСчета(), ПолучитьСчет();`
- `ВыбратьКорСчета(), ПолучитьКорСчет();`
- `ВыбратьВалюты(), ПолучитьВалюту();`
- `ВыбратьПериоды(), ПолучитьПериод();`
- `ВыбратьСубконто(), ПолучитьСубконто();`
- `ВыбратьКорСубконто(), ПолучитьКорСубконто();`

Поддерживаются два режима доступа к соответствующим результатам запроса: перебор всех значений, как показано в листинге 8.3, и поиск по значению, как показано в листинге 8.4.

Листинг 8.3. Перебор счетов

```
БИ.ВыбратьСчета();
Пока БИ.ПолучитьСчет()=1 Цикл
        // Обрабатываем итоги по текущему счету
КонецЦикла;
```

Листинг 8.4. Поиск в выборке счета по значению

```
Если БИ.ПолучитьСчет(,СчетПоКоду("41.1"))=1 Тогда
        // Обрабатываем итоги по счету 41.1
КонецЕсли;
```

Для обращения к результатам бухгалтерского запроса используются атрибуты и функции (`БИ` — переменная типа `"БухгалтерскиеИтоги"`):

- `БИ.Счет` — текущий счет запроса;
- `БИ.Субконто(<ВидСубконто>)` — текущее субконто запроса;
- `БИ.КорСчет` — текущий корреспондирующий счет;
- `БИ.КорСубконто(<ВидСубконто>)` — текущее корреспондирующее субконто запроса;
- `БИ.Валюта` — текущая валюта запроса;
- `БИ.НачДата` — начальная дата текущего периода;
- `БИ.КонДата` — конечная дата текущего периода;
- `БИ.Операция` — текущая операция запроса.

Сальдо:

- `БИ.СНД(<ТипСуммы>)` — сальдо начальное дебетовое;
- `БИ.СКД(<ТипСуммы>)` — сальдо конечное дебетовое;
- `БИ.СНК(<ТипСуммы>)` — сальдо начальное кредитовое;
- `БИ.СКК(<ТипСуммы>)` — сальдо конечное кредитовое.

Обороты:

- `БИ.ДО(<ТипСуммы>)` — дебетовый оборот по текущему счету, текущему субконто;
- `БИ.КО(<ТипСуммы>)` — кредитовый оборот по текущему счету, текущему субконто;

- `БИ.КорДО(<ТипСуммы>)` — дебетовый оборот по текущему корсчету, текущему корсубконто;
- `БИ.КорКО(<ТипСуммы>)` — кредитовый оборот по текущему корсчету, текущему корсубконто.

Рассмотрим несколько примеров использования бухгалтерского запроса.

Оборотно-сальдовая ведомость

Информация для отчета "Оборотно-сальдовая ведомость" (см. табл. 2.1) может быть получена бухгалтерским запросом, как показано в листинге 8.5.

Листинг 8.5. Оборотно-сальдовая ведомость

```
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ВыполнитьЗапрос(НачДата,КонДата);
БИ.ВыбратьСчета();
Пока БИ.ПолучитьСчет()=1 Цикл
        Сообщить(БИ.Счет);
        Сообщить("Сальдо на начало по дебету "+БИ.СНД("С")+
                " по кредиту "+БИ.СНК("С"));
        Сообщить("Оборот по дебету "+БИ.ДО("С")+
                " по кредиту "+БИ.КО("С"));
        Сообщить("Сальдо на конец по дебету "+БИ.СКД("С")+
                " по кредиту "+БИ.СКК("С"));
КонецЦикла;
```

Анализ счета

Отчет "Анализ счета" показывает начальное и конечное сальдо по выбранному счету `ВыбСчет` и его обороты с другими счетами. Информация для отчета "Анализ счета" может быть получена бухгалтерским запросом, как показано в листинге 8.6.

Листинг 8.6. Анализ счета

```
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ВыполнитьЗапрос(НачДата,КонДата,ВыбСчет,,,3);
Если БИ.ПолучитьСчет(,ВыбСчет)=1 Тогда
        Сообщить("Анализ счета " + ВыбСчет);
        Сообщить("Сальдо начальное по дебету "+БИ.СНД("С")+
```

```
            " по кредиту "+БИ.СНК("С"));
        БИ.ВыбратьКорСчета();
        Пока БИ.ПолучитьКорСчет()=1 Цикл
            ТекКорСчет=БИ.КорСчет;
            Сообщить("Оборот со счетом " + ТекКорСчет +
            " по дебету "+БИ.КорДО("С")+" по кредиту "+БИ.КорКО("С"));
        КонецЦикла;
        Сообщить("Сальдо конечное по дебету "+БИ.СКД("С")+
            " по кредиту"+БИ.СКК("С"));
КонецЕсли;
```

Оборотно-сальдовая ведомость по счету

Отчет "Оборотно-сальдовая ведомость по счету" показывает начальное и конечное сальдо, обороты по выбранному счету с детализацией по субконто. Информация для отчета "Оборотно-сальдовая ведомость по счету" может быть получена бухгалтерским запросом, как показано в листинге 8.7.

Листинг 8.7. Оборотно-сальдовая ведомость по счету 41.1 с детализацией по номенклатуре

```
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ИспользоватьСубконто("Номенклатура");
БИ.ВыполнитьЗапрос(НачДата,КонДата,"41.1");
БИ.ВыбратьСубконто(1);
Пока БИ.ПолучитьСубконто(1)=1 Цикл
    Сообщить("Товар "+БИ.Субконто(1).Наименование);
    Сообщить("Остаток на начало равен " + Строка(БИ.СНД("К2")) + 2ед.
    на сумму " + Строка(БИ.СНД("С")));
    Сообщить("Остаток на конец равен " + Строка(БИ.СКД("К")) + "ед. на
    сумму " + Строка(БИ.СКД("С")));
КонецЦикла;
```

Обороты между счетами с детализацией по субконто

Давайте попробуем решить следующую задачу. Нужно определить обороты между счетом 41.1 и 60.1 с детализацией по номенклатуре и контрагентам. Решение этой задачи приведено в листинге 8.8.

Листинг 8.8. Обороты между счетами с детализацией по субконто

```
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ИспользоватьСубконто("Номенклатура");
```

```
БИ.ИспользоватьКорСубконто("Контрагенты");
БИ.ВыполнитьЗапрос(НачДата,КонДата, "41.1", "60.1",,3);
БИ.ВыбратьСубконто(1);
Пока БИ.ПолучитьСубконто(1)=1 Цикл
        Товар=БИ.Субконто(1);
        БИ.ВыбратьКорСубконто(1);
        Пока БИ.ПолучитьКорСубконто(1)=1 Цикл
        Поставщик=БИ.КорСубконто(1);
                Сообщить("Поставщик " +Поставщик +
                " поставил товар "+Товар+
                " на сумму " + БИ.КорДО("С"));
        КонецЦикла;
КонецЦикла;
```

Итоги по валютным счетам

При извлечении информации с валютного счета надо учитывать, что на валютном счете ведется раздельный учет по каждой валюте. Например, пусть на счете 52 имеется сумма в 100 USD и 100 EURO. Тогда, чтобы получить эти суммы с помощью бухгалтерского запроса, нужно перебрать все валюты в цикле, как показано в листинге 8.9.

Листинг 8.9. Получение информации с валютного счета

```
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ВыполнитьЗапрос(НачДата,КонДата, "52");
БИ.ВыбратьВалюты();
Пока БИ.ПолучитьВалюту()=1 Цикл
        Сообщить("На счете имеется остаток "+БИ.СКД("В")+ " "+БИ.Валюта);
КонецЦикла;
```

Промежуточные итоги

Для получения промежуточных итогов нужно задать параметр бухгалтерского запроса <Периодичность> и организовать цикл по периодам, как показано в листинге 8.10.

Листинг 8.10. Получение информации с детализацией по периодам

```
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ВыполнитьЗапрос('01.01.2005','31.12.2005', "41.1,,,, "Месяц");
```

```
БИ.ВыбратьПериоды ();
Пока БИ.ПолучитьПериод()=1 Цикл
        Сообщить("За период с " + БИ.НачДата + " по " + БИ.КонДата +
                "поступило товаров на сумму " + БИ.ДО("С"));
КонецЦикла;
```

Извлечение проводок и операций

С помощью бухгалтерского запроса можно извлечь информацию с детализацией до операции и проводки. Для этого параметр запроса <Периодичность> должен принимать значения "Операция" или "Проводка".

Решим задачу 3 из задания 7.3 с использованием бухгалтерского запроса. Будем считать, что договор займа завершен, если имеется проводка вида Д91.2-К66.4 или Д91.2-К66.44. Ссылку на договор будем получать из третьего субконто кредита найденной проводки. Договор будем считать завершенным досрочно, если дата проводки меньше запланированной даты окончания договора. Текст отчета приведен в листинге 8.11.

Листинг 8.11. Отчет по договорам займа, завершенным до истечения установленного срока

```
Таб=СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("Отчет");
Таб.ВывестиСекцию("Шапка");
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ВыполнитьЗапрос(НачДата,КонДата, "66.4,66.44","91.2",,,"Проводка");
БИ.ВыбратьПериоды();
Пока БИ.ПолучитьПериод()=1 Цикл
        Оп=БИ.Операция;
        Если (Оп.Дебет.Счет=СчетПоКоду("91.2"))
                и ((Оп.Кредит.Счет=СчетПоКоду("66.4"))
                или (Оп.Кредит.Счет=СчетПоКоду("66.44")))  Тогда
                Договор=Оп.Кредит.Субконто(3);
                Если Договор.ДатаОкончания>Оп.ДатаОперации Тогда
                        ПечКлиент=Договор.Заимодавец;
                        ПечАгент=Договор.Агент;
                        ПечДоговор=Договор.НомерДок;
                        ПечДатаНачала=Договора.ДатаНачала;
```

```
                    ПечСрокДоговора=Договор.ДатаОкончания-
                    Договор.ДатаНачала;
                    ПечФактДатаЗавершения=Оп.ДатаОперации;
                    ПечФактСрокЗайма=Оп.ДатаОперации-Договор.ДатаНачала;
                    ПечСумма=Договор.Сумма;
                    Если Оп.Кредит.Счет=СчетПоКоду(«66.4») Тогда
                     ПечПроцент=Оп.Сумма;
                     Иначе
                            ПечПроцент=Оп.ВалСумма;
                     КонецЕсли;
                     ПечВалюта=Договор.Валюта;
                     Таб.ВывестиСекцию("Договор");
              КонецЕсли;
       КонецЕсли;
КонецЦикла;
Таб.Показать();
```

Задание 8.1. Отчет о задержке

В конфигурации, разработанной в предыдущих главах, создайте следующий отчет с помощью запроса по бухгалтерским итогам.

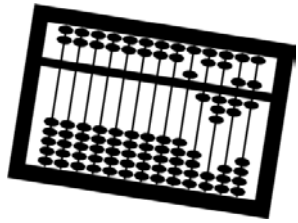
Отчет о задержке в поступлении денежных средств
по заключенным договорам займа
в период с . . . по . . .

Заимодавец	Агент	Договор	Дата начала договора	Дата фактического поступления денежных средств	Сумма договора	Валюта договора	Процент срочный	Процент досрочный
АКМ	ОГЦБ	ДЗ-64/3	12.03.02	17.03.02	1000	US$		

В отчет включаются сведения о договорах, по которым денежные средства поступили с опозданием, то есть позже даты начала договора. Если на дату составления отчета денежные средства по договору так и не поступили, то необходимо в графу **Дата фактического поступления денежных средств** поместить запись "Отсутствует".

8.5. Контрольные вопросы

1. Какова периодичность бухгалтерских итогов?
2. Какая информация хранится в бухгалтерских итогах?
3. Какие режимы работы с бухгалтерскими итогами бывают?
4. Что такое развернутое сальдо?
5. Можно ли получить сальдо по оборотному субконто?
6. Какие атрибуты и функции есть у бухгалтерских итогов в режиме запроса?
7. Как выполнить запрос на позицию документа?



Глава 9

Реализация некоторых учетных алгоритмов

В данной главе мы рассмотрим реализацию некоторых, часто встречающихся на практике, алгоритмов с помощью объектов компоненты "Бухгалтерский учет".

Отметим, какие правила нужно соблюдать при реализации дополнительных возможностей в типовой конфигурации "БухгалтерскийУчет".

1. Внесенные изменения не должны нарушать работу других объектов типовой конфигурации. Предельно осторожно надо выполнять настройку плана счетов. Удалять какие-либо объекты не рекомендуется, так как они могут быть задействованы в других объектах.
2. Нежелательно дублировать существующие в типовой конфигурации функции. Например, в типовой конфигурации есть документ "Выписка", который оформляет *все* движения по расчетному счету. Поэтому добавлять документ типа "Поступление денежных средств по договору займа" не требуется.
3. Надо помнить, что движения по счетам могут выполняться как документами, так и "ручными" операциями. Поэтому для получения бухгалтерских итогов не используйте выборки данных из документов — работайте или с проводками, или с итогами (с помощью объекта `"БухгалтерскиеИтоги"`).
4. При проведении документа бухгалтерская информация, необходимая для расчета суммы проводки, должна извлекаться не из справочников и документов, а из бухгалтерских итогов. Например, при расчете процентов по договору займа нужно ориентироваться не на сумму, указанную в договоре, а на сумму, фактически поступившую на расчетный счет.

9.1. Инвентаризация

Напомним, что *инвентаризация* — это сверка учетного остатка и фактического наличия объектов учета *(см. главу 2)*, которая проводится регулярно с целью проверки данных бухгалтерского учета.

9.1.1. Постановка задачи

Рассмотрим задачу, текст которой приведен в табл. 9.1.

Таблица 9.1. *Инвентаризационная ведомость*

<table>
<tr><th>№</th><th>Текст задания</th><th>Комментарий</th></tr>
<tr><td>1</td><td>Необходимо добавить в типовую конфигурацию отчет следующей формы</td><td>За основу берем текущий релиз конфигурации "Бухгалтерский учет"</td></tr>
<tr><td>2</td><td colspan="2">Инвентаризационная ведомость материалов по складу <склад> на <дата>
<table>
<tr><th>№ п/п</th><th>Наименование материала, ед. измерения</th><th>Учетная цена</th><th>Учетный остаток</th><th>Фактический остаток</th><th>Отклонение (кол-во)</th><th>Отклонение (сумма)</th></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>1</td><td>Шерсть, кг</td><td>20.00</td><td>10</td><td>8</td><td>–2</td><td>– 40.00</td></tr>
<tr><td>2</td><td>Картон, лист</td><td>0.10</td><td>0</td><td>4</td><td>+4</td><td>+ 0.40</td></tr>
<tr><td>3</td><td>Тесьма, м</td><td>1.10</td><td>12</td><td>15</td><td>+3</td><td>+ 3.30</td></tr>
</table></td></tr>
<tr><td>3</td><td>В отчет включить все материалы, учитываемые на всех субсчетах счета 10, в том числе имеющие нулевые остатки по количеству и/или стоимости</td><td>Синтетический счет 10 "Материалы" делится на субсчетах по видам материалов: 10.1 — "Сырье и материалы", 10.2 — "Покупные полуфабрикаты", 10.3 — "Топливо" и т. д.

Особенностью учета материалов, реализованного в типовой конфигурации, является то, что по первому субконто "Материалы" ведется суммовой и количественный учет, а по второму субконто "МестаХранения" — только количественный учет.

У каждого элемента справочника "Материалы" есть привязка к субсчету, на котором он учитывается, — это реквизит "Субсчет10". В результате часто происходит путаница, когда в справочнике указан один субсчет, а остатки находятся на другом субсчете.</td></tr>
</table>

Таблица 9.1 *(продолжение)*

№	Текст задания	Комментарий
		Требование включить в отчет материалы с нулевым количеством или стоимостью обусловлено тем, что на складе могут быть излишки. Однако нет смысла включать в отчет те материалы, которые никогда на этот склад не приходили. Отметим также, что определить стоимость материалов на складе мы можем только расчетным путем, перемножив количественный остаток на среднюю цену по всем складам. Поэтому в нашем случае нулевое количество будет автоматически давать нам нулевую стоимость
4	Значение учетной цены (графа 3) определяется из справочника материалов	Учет материалов в типовой конфигурации реализован по средневзвешенной цене по всем местам хранения. Поэтому списание товара логично было бы выполнять по средней цене. А для оприходования излишков действительно можно взять стоимость из справочника, так как она обычно совпадает с ценой последнего прихода
5	Учетный остаток (графа 4) представлен в натуральном выражении и должен соответствовать данным учета на дату формирования отчета	Ничего не сказано про время: нужно получать отчет на начало дня выбранной даты или на конец дня? Будем получать на конец дня
6	В отчет включаются данные только по одному из складов, который определяется пользователем в диалоге	Так как на ряде субсчетов счета 10 отсутствует субконто "МестаХранения", то эти субсчета мы автоматически исключаем из отчета
	Необходимо предусмотреть следующие режимы работы с отчетом	
7	**Сформировать** — формируется отчет путем заполнения граф 1, 2, 3, 4	Формировать отчет будем с помощью бухгалтерского запроса, с использованием двух субконто: "Материалы" и "МестаХранения", но без отбора, так как нам нужно будет получать среднюю цену по материалу по всем складам
8	**Ввод фактических остатков** — пользователь непосредственно в отчет вводит вручную фактические остатки в натуральном выражении (графа 5)	Для ввода данных у нас есть несколько способов *(см. главу 6)*. Самый простой, на наш взгляд, способ — использовать пустую таблицу

Таблица 9.1 *(окончание)*

№	Текст задания	Комментарий
9	**Расчет** — рассчитывается величина отклонения в натуральном и стоимостном выражении, результаты заносятся соответственно в графы 6 и 7. Отклонение в натуральном выражении (графа 5) рассчитывается как разница между фактическим и учетным остатком (графа 5 – графа 4). Сумма отклонения (графа 7) рассчитывается путем умножения учетной цены (графа 3) на величину отклонения в натуральном выражении (графа 6)	Для расчета отклонений будем оперировать ячейками таблицы с помощью функции `Область()`
10	**Проведение** — формируются корректирующие проводки по каждому материалу, по которому выявлено отклонение в натуральном выражении. Сумма проводки представляет собой абсолютную величину отклонения, указанного в графе 7. В проводке указывается следующая корреспонденция счетов: ***при положительном отклонении***: дебет счета 10 (соответствующий субсчет), кредит счета 91.1 ***при отрицательном отклонении***: кредит счета 10 (соответствующий субсчет), дебет счета 94	Будем создавать "ручную операцию", на основании данных расчета п. 9. Для формирования проводок нужно определить аналитику. По 10-му счету: Материал будем брать из расшифровки, место хранения из реквизита диалога **Склад**. На 94-м счете ("Недостачи от порчи ценностей") аналитики нет. Аналитику по счету 91.1 ("Прочие доходы") будем запрашивать в форме диалога

9.1.2. Анализ и решение задачи

Для решения данной задачи будем использовать бухгалтерский запрос *(см. главу 8)* и пустую таблицу *(см. главу 6)*. Текст программного модуля приведен в листинге 9.1.

Форма диалога

Форма диалога приведена на рис. 9.1. На форме поместим следующие реквизиты:

- `ДатаИнвентаризации` (тип `Дата`) — дата, на конец дня которой производится инвентаризация;
- `Склад` (тип `Справочник.МестаХранения`) — склад, на котором производится инвентаризация;

- `СтатьяДоходов` (тип `Справочник.ПрочиеДоходыИРасходы`) — значение субконто по счету 91.1, в случае оприходования излишков материалов.

Дополнительно к кнопкам **Сформировать** и **Закрыть** поместим кнопки **Рассчитать** (вызывает процедуру `Рассчитать()`) и **Провести** (вызывает процедуру `Проведение()`).

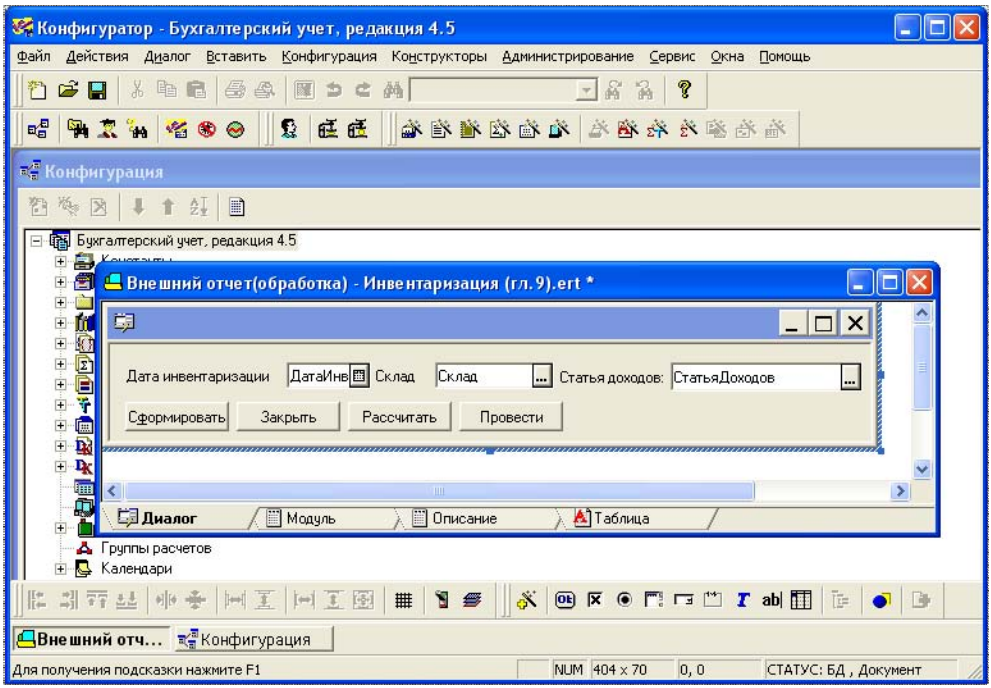


Рис. 9.1. Форма диалога отчета "Инвентаризационная ведомость"

Формирование отчета

Перед выполнением запроса зададим использование видов субконто "Материалы" и "МестаХранения", причем без отборов, так как нам понадобится средняя цена по всем складам.

В задании сказано, что в отчет нужно включать, в том числе, материалы с нулевым остатком. Однако особенностью запроса является то, что в его результат включаются только те итоги, которые имеют ненулевые остатки и/или обороты. Поэтому если мы будем делать запрос только на конечную дату, то материалы с нулевым остатком, но ненулевым оборотом, в результат запроса не попадут. Выберем некоторую дату (начало работы организации, начало года, дата первой проводки и т. д.), с которой будут выбираться бухгалтерские итоги.

Далее, в качестве фильтра по счетам укажем "10". Это означает, что в запрос попадут все проводки по всем субсчетам счета 10. Точнее, не все субсчета, а только те, у которых есть аналитический учет по видам субконто "Материалы" и "МестаХранения" (таким образом, мы автоматически отсекаем данные субсчетов 10.7, 10.11).

Затем перебираем в цикле значения субконто "Материалы" и субсчета счета 10. Почему именно в такой последовательности? Если наоборот, перебирать сначала счета, а затем материалы, то один и тот же материал будет выведен в разных местах отчета, а это неудобно.

Получив текущий остаток по всем складам, мы вычисляем среднюю цену материала, если остаток положительный. В противном случае мы берем цену из карточки материала.

После этого находим в бухгалтерских итогах значение второго субконто, равное значению реквизита "Склад", выбранному в форме диалога.

Внимание!

При использовании функции `ПолучитьСубконто()` искомое значение субконто указывается третьим параметром!

Теперь осталось вывести полученные итоги в таблицу. Таблица-шаблон приведена на рис. 9.2. Помимо текстовой информации, мы будем хранить в ячейках таблицы расшифровку соответствующих значений: `Материал`, `СчетУчета`, `УчетнаяЦена`.

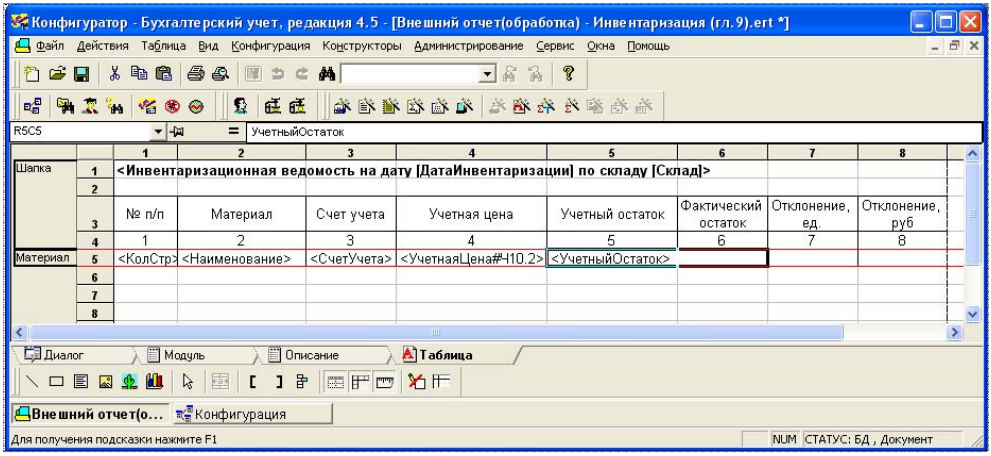


Рис. 9.2. Таблица-шаблон для инвентаризационной ведомости

И последнее замечание, для того чтобы обрабатывать таблицу, нам нужно знать количество выведенных строк. Для этого в программном модуле

определим переменную `КолСтр`, которая будет хранить количество выведенных строк.

Расчет отклонений

Расчет отклонений производится путем манипуляций с ячейками таблицы с помощью функции таблицы `Область()`.

Проведение

При проведении мы формируем "ручную" операцию по указанным в задании счетам.

Как проверить правильность отчета?

Поскольку цель инвентаризации — совпадение учетных и фактических остатков, то после ввода корректирующих проводок и повторного формирования отчета, остатки в колонке **Фактический остаток** должны переместиться в колонку **Учетный остаток**.

Листинг 9.1. Программный модуль отчета "Инвентаризационная ведомость"

```
Перем КолСтр;
//
// Формируем отчет
//
Процедура Сформировать()
Таб=Таблица;
Таб.Очистить();
Таб.ВывестиСекцию("Шапка");
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ИспользоватьСубконто("Материалы");
БИ.ИспользоватьСубконто("МестаХранения");
БИ.ВыполнитьЗапрос('01.01.2000',ДатаИнвентаризации,"10");
КолСтр=0;
БИ.ВыбратьСубконто(1);
Пока БИ.ПолучитьСубконто(1)=1 Цикл
БИ.ВыбратьСчета();
Пока БИ.ПолучитьСчет()=1 Цикл
            Материал=БИ.Субконто(1);
                // Определяем учетную цену
                КолПоВсемСкладам=БИ.СКД("К");
```

```
                СуммаПоВсемСкладам=БИ.СКД("С");
                Если КолПоВсемСкладам>0 Тогда
                        УчетнаяЦена=СуммаПоВсемСкладам/КолПоВсемСкладам;
                        СчетУчета=БИ.Счет;
                        Если СчетУчета<>Материал.Субсчет10 Тогда
Сообщить("Исправьте счет учета в карточке материала "+Материал);
                        КонецЕсли;
                Иначе
                        УчетнаяЦена=Материал.Цена;
                        СчетУчета=Материал.Субсчет10;
                КонецЕсли;
                // Теперь получаем остаток по складу
                Если БИ.ПолучитьСубконто(2,,Склад)=1 Тогда
                        УчетныйОстаток=БИ.СКД("К");
                Иначе
                        УчетныйОстаток=0;
                КонецЕсли;
Наименование=Материал.Наименование+"," +Материал.ЕдиницаИзмерения;
                КолСтр=КолСтр+1;
                Таб.ВывестиСекцию("Материал");
        КонецЦикла;
КонецЦикла;
Таб.Показать();
КонецПроцедуры
//
// Делаем расчет отклонений
//
Процедура Рассчитать()
Таб=Таблица;
Для НомерСтроки=5 По 4+КолСтр Цикл
        УчетнаяЦена=Таб.Область(НомерСтроки,4).Расшифровка();
        УчетныйОстаток=Таб.Область(НомерСтроки,5).Расшифровка();
        ФактическийОстаток=Число(Таб.Область(НомерСтроки,6).Текст);
        РазницаЕд=УчетныйОстаток-ФактическийОстаток;
        РазницаРуб=РазницаЕд*УчетнаяЦена;
        Таб.Область(НомерСтроки,7).Текст=Строка(РазницаЕд);
        Таб.Область(НомерСтроки,7).Расшифровка(РазницаЕд);
        Таб.Область(НомерСтроки,8).Текст=Строка(РазницаРуб);
```

```
        Таб.Область(НомерСтроки,8).Расшифровка(РазницаРуб);
КонецЦикла;
Таб.Показать();
КонецПроцедуры
//
// Формируем проводки
//
Процедура Проведение()
Оп=СоздатьОбъект("Операция");
Оп.Новая();
Оп.ДатаОперации=ДатаИнвентаризации;
Таб=Таблица;
Для НомерСтроки=5 По 4+КолСтр Цикл
        Материал=Таб.Область(НомерСтроки,2).Расшифровка();
        СчетУчета=Таб.Область(НомерСтроки,3).Расшифровка();
        РазницаЕд=Таб.Область(НомерСтроки,7).Расшифровка();
        РазницаРуб=Таб.Область(НомерСтроки,8).Расшифровка();
        Если РазницаЕд>0 Тогда
                Оп.НоваяПроводка();
                Оп.Кредит.Счет=СчетУчета;
                Оп.Кредит.Субконто(1,Материал);
                Оп.Кредит.Субконто(2,Склад);
                Оп.Дебет.Счет=СчетПоКоду("94");
                Оп.Количество=РазницаЕд;
                Оп.Сумма=РазницаРуб
        ИначеЕсли РазницаЕд<0 Тогда
                Оп.НоваяПроводка();
                Оп.Дебет.Счет=СчетУчета;
                Оп.Дебет.Субконто(1,Материал);
                Оп.Дебет.Субконто(2,Склад);
                Оп.Кредит.Счет=СчетПоКоду("91.1");
                Оп.Кредит.Субконто(1,СтатьяДоходов);
                Оп.Количество=-РазницаЕд;
                Оп.Сумма=-РазницаРуб;
        КонецЕсли;
КонецЦикла;
Оп.Записать();
КонецПроцедуры
```

Задание 9.1

1. Напишите аналогичный отчет "Инвентаризационная ведомость товаров по складу", в котором нужно выводить данные по счету 41.1.
2. Недостатком разработанного отчета является то, что первичная информация о фактических остатках теряется вместе с закрытием отчета, а остаются только корректирующие проводки. Чтобы первичная информация об инвентаризации сохранялась, разработайте документ "ИнвентаризацияМатериалов" с аналогичными функциями.

Указание

Если задача вызывает затруднения, посмотрите, как в типовой конфигурации реализован документ "ИнвентаризацияТМЦ".

9.2. Оценка высоколиквидных активов

Высоколиквидными активами называется такое имущество предприятия, которое легко обратить в денежные средства. Таковыми являются сами наличные и безналичные денежные средства в рублях и валюте и ценные бумаги.

9.2.1. Постановка задачи

Необходимо внести дополнения в типовую конфигурацию, включив в нее отчет "Оценка высоколиквидных активов предприятия", который обеспечивает сопоставление денежных активов предприятия в рублевой и валютной оценке на две сопоставляемые даты, задаваемые пользователем в диалоге.

Необходимо в таблицу помещать данные по остаткам соответствующих счетов в рублевой и валютной оценке, при этом должны быть выполнены следующие требования:

1. Валюта, в которой производится оценка активов, а также даты, на которые производится оценка, определяются в диалоге с пользователем.
2. Если на счете имеется остаток в рублях, то в графах 2 и 4 он учитывается без пересчета, а в графах 3 и 5 пересчитывается по курсу валюты, в которой производится оценка на соответствующую дату.
3. Если на счете имеется остаток в валюте, в которой производится оценка, то в графах 3 и 5 он учитывается без пересчета, а в графах 2 и 4 пересчитывается в рубли по курсу валюты, в которой производится оценка на соответствующую дату.

Оценка высоколиквидных активов на "дата1" и "дата2" в рублях и валюте "валюта"

Наименование актива	**Оценка на "дата 1"**		**Оценка на "дата 2"**	
	В рублях	**В "валюта"**	**В рублях**	**В "валюта"**
1	**2**	**3**	**4**	**5**
Наличные денежные средства в рублях (счет 50.1)				
Наличная валюта (счет 50.11)				
На рублевых счетах (счет 51)				
На валютных счетах (счет 52)				
Паи и акции (счет 58.1)				
Долговые ценные бумаги (счет 58.2)				
Предоставленные займы (счет 58.3)				
Паи и акции в валюте (счет 58.11)				
Долговые ценные бумаги в валюте (счет 58.22)				
Предоставленные займы в валюте (счет 58.33)				
Итого денежных средств				

4. Если на счете имеются остатки в валюте, отличной от валюты, в которой производится оценка, то в графах 2 и 4 они учитываются в рублевом эквиваленте, рассчитанном по курсу на требуемую дату, а в графах 3 и 5 пересчитываются в оценочную валюту из соответствующей рублевой оценки.
5. Если на счете имеются остатки, указанные в пп. 3, 4, 5, то в графы 2—5 помещаются суммы всех остатков, причем для каждого из остатков применяются правила пересчета, означенные в соответствующих пунктах.

9.2.2. Анализ задачи и решение

Несмотря на весьма громоздкую постановку задачи, решение получается довольно простым (листинг 9.2).

Форма диалога

В форме диалога (рис. 9.3) поместим реквизиты `НачДата` (тип `Дата`), `КонДата` (тип `Дата`) и `ВалютаОценки` (тип `Справочник.Валюты`).

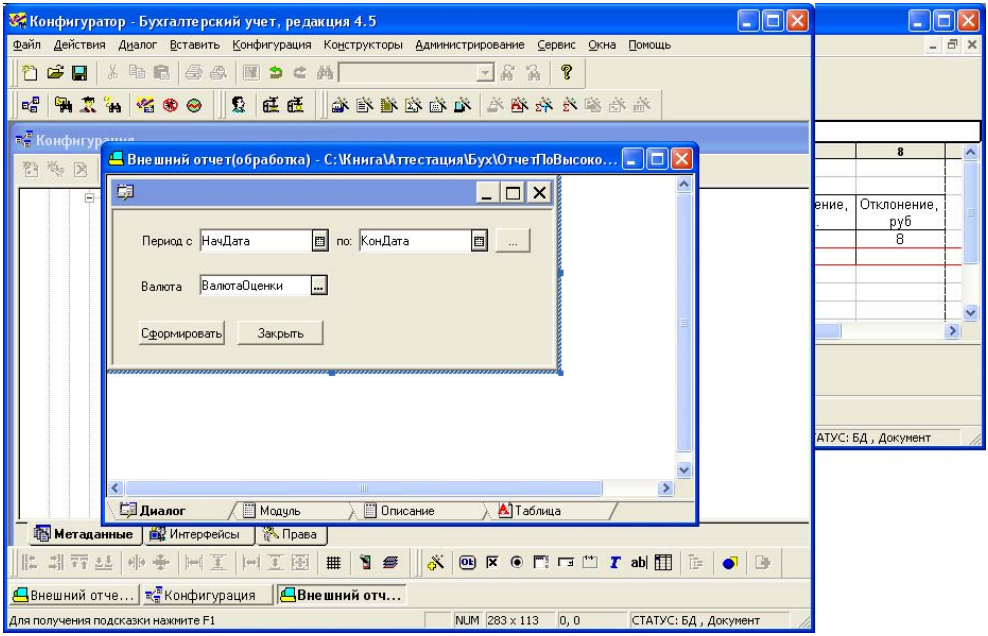


Рис. 9.3. Форма диалога отчета по высоколиквидным активам

Формирование отчета

При решении воспользуемся бухгалтерским запросом с перебором итогов по счетам и валютам *(см. главу 8)*.

Если текущий счет является рублевым, то тогда мы просто извлекаем итоги функциями `СНД()` и `СКД()`. Если счет валютный, то мы перебираем все валюты, имеющиеся на счете, и накапливаем их рублевый эквивалент в переменных `СумР1` и `СумР2`. Полученные рублевые суммы мы выводим в графах "2" и "4", а в графах "3" и "5" пересчитываем эти сумму по курсу валюты оценки.

Листинг 9.2. Отчет по высоколиквидным активам предприятия

```
Процедура Сформировать()
Таб=СоздатьОбъект("Таблица");
Таб.ВывестиСекцию("Шапка");
```

```
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ВыполнитьЗапрос(НачДата, КонДата, "50,51,52,58");
БИ.ВыбратьСчета();
Пока БИ.ПолучитьСчет()=1  Цикл
        Счет=БИ.Счет;
        Если Счет.Валютный=1 Тогда
                СумР1=0; СумР2=0;
                БИ.ВыбратьВалюты();
                Пока БИ.ПолучитьВалюту()=1 Цикл
                        Валюта=БИ.Валюта;
                        СумР1=СумР1+БИ.СНД("В")*
                                Валюта.Курс.Получить(НачДата)/
                                Валюта.Кратность.Получить(НачДата);
                        СумР2=СумР2+БИ.СКД("В")*
                                Валюта.Курс.Получить(КонДата)/
                                Валюта.Кратность.Получить(КонДата);
                КонецЦикла;
        Иначе
                СумР1=БИ.СНД("С");
                СумР2=БИ.СКД("С");
        КонецЕсли;
        СумВ1=СумР1/ВалютаОценки.Курс.Получить(НачДата)*
                        ВалютаОценки.Кратность.Получить(НачДата);
        СумВ2=СумР2/ВалютаОценки.Курс.Получить(КонДата)*
                        ВалютаОценки.Кратность.Получить(КонДата);
        Таб.ВывестиСекцию("Строка");
КонецЦикла;
Таб.Показать();
КонецПроцедуры
```

Задание 9.2

1. Оформите отчет, включив в него логотип предприятия.
2. Предусмотрите вызов созданного отчета через меню.
3. Обеспечьте возможность обновления отчета и детализацию показателей отчета по аналитике, валютам и пр.

9.3. Учет ценных бумаг

Ценные бумаги — векселя, акции, паи и т. д. — относятся к краткосрочным финансовым вложениям и учитываются на счете 58. Для ценных бумаг, которые котируются на фондовом рынке, характерно частое изменение цены, вследствие чего в целях бухгалтерского учета возможно применение различных алгоритмов учета их себестоимости:

- по средневзвешенной цене;
- FIFO: первый пришел — первый ушел;
- LIFO: последний пришел — первый ушел.

Для последних двух алгоритмов необходимо учитывать остатки ценных бумаг по каждой партии прихода ценных бумаг. Такой учет называют *партионным*.

Рассмотрим следующий пример (табл. 9.2). Ценные бумаги приобретаются несколькими партиями, причем новая партия с другой ценой может поступить еще до момента полной продажи ценных бумаг предыдущей партии.

Таблица 9.2. *Покупка и продажа ценных бумаг*

Дата	Приход, количество и цена	Расход, количество	Себестоимость по среднему	FIFO	LIFO
01.01	10 × 10 р.				
02.01	10 × 20 р.				
03.01	10 × 30 р.				
04.01		10	200	100	300
05.01		10	200	200	200
06.01		10	200	300	100
Итого	= 600		= 600	= 600	= 600

Естественно, что в результате общая себестоимость по всем алгоритмам совпадает, но в отдельные моменты времени себестоимость реализации и прибыль (убыток) от реализации ценных бумаг будут отличаться.

9.3.1. Постановка задачи

Текст задачи по учету ценных бумаг приведен в табл. 9.3.

Таблица 9.3. *Текст задания с комментариями*

№	Текст задания	Комментарий
1	Необходимо внести дополнения в типовую конфигурацию для реализации учета ценных бумаг на счете 58 "Краткосрочные финансовые вложения"	За основу берем текущий релиз конфигурации "Бухгалтерский учет"
2	Для этого необходимо создать объекты, содержащие следующую информацию: **Сведения о контрагентах** — организации или физические лица, осуществляющие покупку или продажу ЦБ; **Сведения об агентах** — структурное подразделение или отв. лицо (сотрудник), осуществляющее сделку с ЦБ; **Сведения о заключенных договорах купли-продажи ЦБ** – документ, являющийся основанием совершения операций купли-продажи; **Сведения о ценных бумагах** — вид ценной бумаги, эмитент, тип ценной бумаги, номер выпуска, номинальная стоимость; **Виды ценных бумаг:** акции, облигации, векселя и т. д.; **Типы ценных бумаг:** обыкновенные, привилегированные, с купонным доходом и т. п.; **Сведения об эмитентах:** наименование и реквизиты организации-эмитента (Минфин, Газпром, Лукойл, Сбербанк и т. п.)	Информацию о контрагентах и агентах будем хранить в уже существующем справочнике "Контрагенты". Для хранения информации о договоре купли-продажи создадим документ "ДоговорКупли**Продажи**ЦБ". Сведения о ценных бумагах будем хранить в уже существующем справочнике "ЦенныеБумаги". Для видов и типов ценных бумаг создадим перечисления, а информацию об эмитентах будем хранить в уже существующем справочнике "Контрагенты"
3	*В рабочем плане счетов* на счете 58 открыть субсчет для учета ценных бумаг в количественном и стоимостном выражении, предусмотреть ведение на нем аналитического учета в двух независимых разрезах: ценных бумаг и агентов	Заведем субсчет 58.10. На 58-м счете уже ведется учет в разрезе контрагентов, поэтому этот вид субконто оставляем, но будем записывать туда агентов, а добавляем второй вид субконто "ЦенныеБумаги". На счете 58.10 надо установить признак ведения количественного учета

Таблица 9.3 *(продолжение)*

№	Текст задания	Комментарий
4	Для расчетов с покупателями и продавцами ценных бумаг выделяется субсчет на счете 76, который ведется в аналитическом разрезе контрагентов и договоров купли-продажи ЦБ	Заведем субсчет 76.10. На нем определяем два вида субконто "Контрагенты" и "ДоговорыЦБ". Вид субконто "ДоговорыЦБ" нужно предварительно создать (тип `Документ.ДоговорКупли-ПродажиЦБ`)
5	*Предусмотреть ввод и хранение первичных документов* "`Договор купли-продажи ЦБ`", имеющего следующие реквизиты: номер договора; дата заключения договора; агент; контрагент, операция (покупка/продажа); Табличные: ценная бумага, количество, цена, сумма. Разработайте документы для автоматизации ввода следующих операций: **Заключение договора** — ввод первичного документа "`Договор купли-продажи ЦБ`" в базу данных бухгалтерского учета	Добавляем новый вид документов "`ДоговорКупли-ПродажиЦБ`" с реквизитами шапки: • `Агент` (тип `Справочник.Контрагенты`) • `Контрагент` (тип `Справочник.Контрагенты`) • `ОперацияЦБ` (тип `Перечисление.ВидыОпераций-ЦБ`, которое нужно предварительно создать) и реквизитами табличной части: • `ЦеннаяБумага` (тип `Справочник.ЦенныеБумаги`) • `Количество` (тип `Число`) • `Цена` (тип `Число`) • `Сумма` (тип `Число`)
6	**Покупка ЦБ** (по договору покупки): **Оплата.** На основании банковской выписки, подтверждающей факт произведенной оплаты, формируется запись в дебет счета 76 с кредита счета денежных средств на сумму оплаты. **Приход ЦБ**. При получении выписки со счета ДЕПО производятся записи в дебет счета 58 с кредита счета 76 – приход ЦБ по фактической (договорной) стоимости по каждой ценной бумаге.	"Выписка со счета ДЕПО" — первичный (бумажный) документ, подтверждающий факт перехода права собственности на ценную бумагу от продавца к покупателю. Расход денег с расчетного счета осуществляется с помощью стандартного документа типовой конфигурации "Выписка". Для отражения факта прихода ценных бумаг создадим документ "`ПриходЦБ`"

Таблица 9.3 *(окончание)*

№	Текст задания	Комментарий
7	**Реализация ЦБ** (по договору продажи): **Оплата.** На расчетный счет предприятия или в кассу поступают денежные средства в счет оплаты ЦБ по договорной цене. На основании платежных документов формируется запись в кредит счета 76 с дебета счета денежных средств на сумму оплаты. **Реализация ЦБ.** При получении выписки со счета ДЕПО производятся записи с кредита счета 91 в дебет счета 76 на сумму выручки от реализации ЦБ по фактической (договорной) стоимости по каждой ценной бумаге. С кредита счета 58 в дебет счета 91 по каждой ценной бумаге делается запись на сумму, определенную исходя из учетной стоимости ЦБ (по методу LIFO, FIFO или средневзвешенной цене)	Поступление денег на расчетный счет осуществляется с помощью стандартного документа типовой конфигурации "Выписка". Для отражения факта расхода ценных бумаг создадим документ "`РасходЦБ`"
8	Обеспечьте, чтобы при изменении учетных данных "задним числом" программа автоматически определяла приходные и/или расходные документы, нуждающиеся в повторном проведении для правильного списания учетной стоимости ЦБ. Такие документы должны автоматически получать статус "непроведенный"	Эти действия будем делать при записи документов "`ПриходЦБ`" и "`РасходЦБ`"

9.3.2. Реализация алгоритма "По среднему"

Алгоритм "По среднему" реализуется наиболее просто: определяется текущая стоимость ценной бумаги и текущий остаток. Их отношение дает нам среднюю цену. Перемножая среднюю цену на реализуемое количество, получаем себестоимость. Алгоритм приведен в листинге 9.3.

Листинг 9.3. Алгоритм вычисления средневзвешенной себестоимости

```
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ИспользоватьСубконто("ЦенныеБумаги",ЦБ,2);
БИ.ВыполнитьЗапрос(,ТекущийДокумент(),"58.10");
Сум=БИ.СКД("С");
Ост=БИ.СКД("К");
Если Ост=Количество Тогда
        СумСпис=Сум;
ИначеЕсли Ост>Количество Тогда
        ЦенаСпис=Сум/Ост;
        СумСпис=Окр(Количество*ЦенаСпис,2);
Иначе
        Сообщить("Остаток меньше нуля");
        НеПроводитьДокумент();
КонецЕсли;
Операция.НоваяПроводка();
Операция.Дебет.Счет=СчетПоКоду("91.2");
Операция.Кредит.Счет=СчетПоКоду("58.10");
Операция.ЦенныеБумаги=ЦБ;
Операция.Сумма=СумСпис;
Операция.Количество=Количество;
```

Так как при определении средней цены возможны ошибки округления, то при списании остатка нужно списывать всю оставшуюся сумму.

9.3.3. Реализация алгоритмов FIFO и LIFO

Для реализации алгоритмов FIFO и LIFO необходимо добавить на счет учета ценных бумаг субконто "Партии" (тип Документ.ПриходЦБ). При поступлении ценных бумаг в модуле документа "ПриходЦБ" выполним привязку к партии, как показано в листинге 9.4.

Листинг 9.4. Модуль документа "ПриходЦБ"

```
Процедура ОбработкаПроведения()
ВыбратьСтроки();
Пока ПолучитьСтроку()=1 Цикл
```

```
Операция.НоваяПроводка();
Операция.Дебет.Счет=СчетПоКоду("58.10");
Операция.Дебет.Субконто(1,Агент);
Операция.Дебет.Субконто(2,ЦеннаяБумага);
Операция.Дебет.Субконто(3,ТекущийДокумент());
Операция.Кредит.Счет=СчетПоКоду("76.10");
Операция.Кредит.Субконто(1,Контрагент);
Операция.Кредит.Субконто(2,ДоговорКуплиПродажиЦБ);
Операция.Количество=Количество;
Операция.Сумма=Сумма;
КонецЦикла;
Операция.Записать();
КонецПроцедуры
```

Алгоритм расчета себестоимости по методам FIFO и LIFO в модуле документа `"РасходЦБ"` приведен в листинге 9.5. С помощью бухгалтерского запроса извлекаем и перебираем остатки по партиям.

Листинг 9.5. Алгоритмы FIFO и LIFO

```
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
БИ.ИспользоватьСубконто("ЦенныеБумаги",ЦеннаяБумага,2);
БИ.ИспользоватьСубконто("ПартииЦБ");
БИ.ВыполнитьЗапрос(,ТекущийДокумент(),"58.10");
Если Константа.МетодОпределенияСебестоимости =
Перечисление.МетодыОпределенияСебестоимости.ФИФО Тогда
        // Выбираем партии в порядке возрастания партий
        БИ.ВыбратьСубконто(2,,,,,0);
Иначе
        // Выбираем партии в порядке убывания партий
        БИ.ВыбратьСубконто(2,,,,,1);
КонецЕсли;
НераспределенноеКол=Количество;
Пока БИ.ПолучитьСубконто(2)=1 Цикл
        Сум=БИ.СКД("С");
        Ост=БИ.СКД("К");
        Если Ост<= НераспределенноеКол Тогда
                // Списываем весь остаток по партии
```

```
            СумСпис=Сум;
            КолСпис=Ост;
        ИначеЕсли Ост> НераспределенноеКол Тогда
            // Списываем требуемое количество
            ЦенаСпис=Сум/Ост;
            КолСпис= НераспределенноеКол;
            СумСпис=Окр(КолСпис*ЦенаСпис,2);
        Иначе
            Продолжить;
        КонецЕсли;
        Операция.НоваяПроводка();
        Операция.Дебет.Счет=СчетПоКоду(«91.2»);
        Операция.Кредит.Счет=СчетПоКоду(«58.10»;
        Операция.Кредит.ЦенныеБумаги=ЦБ;
        Операция.Кредит.ПартииЦБ=БИ.Субконто(2);
        Операция.Сумма=СумСпис;
        Операция.Количество=КолСпис;
        НераспределенноеКол = НераспределенноеКол – КолСпис;
        Если НераспределенноеКол = 0 Тогда
            Прервать;
        КонецЕсли;
КонецЦикла;
Если НераспределенноеКол >0 Тогда
        Сообщить("Остаток меньше нуля");
        НеПроводитьДокумент();
КонецЕсли;
```

Задание 9.3

Обеспечьте формирование следующих отчетов:

1. Портфель ценных бумаг

Наименование ЦБ	Количество	Номинальная стоимость	Учетная цена	Учетная стоимость
1	2	3	4	5

Отчет должен содержать информацию о наличии на предприятии ценных бумаг.

2. Отчет о реализации ценных бумаг за период с ... по ...

Наименование ЦБ	Агент	Количество реализовано	Выручка от реализации

Отчет должен содержать информацию о реализации ценных бумаг за период, указанный пользователем.

3. Отчет о приобретении ценных бумаг за период с ... по ...

Наименование ЦБ	Агент	Количество	Сумма

Отчет должен содержать информацию о поступлении ценных бумаг за период, указанный пользователем.

9.3.4. Отмена проведения документов

Поскольку при изменении документов задним числом нарушается правильный расчет себестоимости ценных бумаг, то в данной задаче предлагается отменять проведение последующих документов. В форме документов "ПриходЦБ" и "РасходЦБ" опишем предопределенную процедуру ПриЗаписи(), как показано в листинге 9.6. Аналогичные действия нужно предусмотреть при отмене проведения и при пометке документа на удаление.

В *главе 10* будет рассмотрено понятие последовательности документов, с помощью которой эта задача решается более корректно.

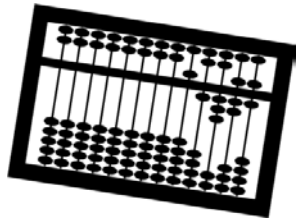
Листинг 9.6. Отмена проведения документов

```
Процедура ПриЗаписи()
Записать();
Док=СоздатьОбъект("Документ");
Док.ВыбратьДокументы(ТекущийДокумент(),);
Пока Док.ПолучитьДокумент()=1 Цикл
        Если Док.ТекущийДокумент()<>ТекущийДокумент() Тогда
        Если (Док.Вид()= "ПриходЦБ") или (Док.Вид()= "РасходЦБ") Тогда
                Док.СделатьНеПроведенным();
```

```
        КонецЕсли;
        КонецЕсли;
КонецЦикла;
КонецПроцедуры
```

9.4. Контрольные вопросы

1. Что такое инвентаризация? Может ли инвентаризация проводиться одновременно с работой склада?
2. Как реализовать получение инвентаризационной ведомости по всем материалам, независимо от того, поступали они или нет?
3. Почему нельзя оценивать высоколиквидные активы в иностранной валюте с помощью функций `СумР=БИ.СНД("С")` и `СумР=БИ.СКД("С")`?
4. Какие признаки должны быть установлены у субконто "Агенты" на счете 58?



Глава 10

Оперативный учет

Для ведения оперативного учета в 1С:Предприятии 7.7 предназначена компонента "Оперативный учет", которая предоставляет несколько специализированных механизмов. Чтобы понять, что такое оперативный учет, рассмотрим два способа ведения учета:

- *неоперативный учет, или работа "задним числом"* — первичные документы, по которым ведется учет, накапливаются "в стопочку". В конце отчетного периода бухгалтер садится за компьютер и вводит их в программу "задним числом". Потом формируются отчеты и сдаются в налоговую инспекцию (подаются руководителю). Такой способ ведения учета достаточно распространен, особенно когда бухгалтер вынужден выполнять параллельно множество других обязанностей. Основной недостаток данного подхода в том, что объективная информация о деятельности предприятия оказывается известной только в моменты сдачи отчетности. Кроме того, при вводе данных "задним числом" приходится вспоминать особенности тех или иных операций;
- *оперативный учет или работа в "реальном времени"* — первичные документы вводятся в компьютер одновременно с совершением хозяйственной операции. Особенно это удобно, когда выписка первичного документа производится не "от руки", а на компьютере с выводом печатной формы на принтере. В этом случае можно организовать оперативный учет, при котором состояние организации можно оценить в любой момент времени (оперативно). Это действительно удобно, когда при выписке счета или накладной менеджер знает текущие остатки на складе, когда известна текущая сумма долга клиента, и мгновенно можно узнать, поступила ли оплата по выставленному счету. Бухгалтер может оперативно оценить сумму НДС к уплате, а руководитель — сумму товарооборота и полученную прибыль.

В реальной организации обычно присутствуют и оперативный, и неоперативный учет. Наиболее важные участки учета ведутся оперативно, а то, что может подождать, — подождет до момента сдачи отчета.

Проблема организации оперативного учета в небольших фирмах заключается в недостатке ресурсов (техника и люди), а в крупных — в масштабе деятельности.

ПРИМЕРЫ

Даже такой простой вопрос, как учет остатков товаров по складу, может вызывать существенные проблемы.

Ситуация 1. Товар на складе есть — по компьютеру нет. Документы приходят и вводятся в учетную систему позже их реального поступления на склад. В результате приходится отключать контроль отрицательных остатков товаров на складе, чтобы отгрузить товар покупателю, и ситуация выходит из-под контроля.

Ситуация 2. Товар по компьютеру есть — на складе нет. Самый простой вариант — *пересортица*. По компьютеру провели одну номенклатуру товара, реально отгрузили другую, похожую. В результате по одной номенклатуре возникает излишек товара, по другой — недостача. Другие варианты: ошибка ввода данных в систему, изменение характеристик товара в процессе хранения (например, товар высыхает), ошибка измерения количества товара (измерение веса производится на разных весах) и т. д.

Ситуация 3. Работа в распределенной информационной базе. Поскольку синхронизация данных на удаленных рабочих местах производится с некоторым шагом по времени, то система на одном удаленном рабочем месте может "не знать" о наличии товара, так как информация вводится с другого удаленного места. Эти и другие подобные ситуации не всегда удается разрешить чисто техническими средствами — приходится принимать различные организационные меры.

Технические ресурсы включают в себя компьютерную технику, программное обеспечение, различные устройства для обеспечения быстрого ввода информации: сканеры штрихкодов, электронные весы, контрольно-кассовые машины (ККМ), магнитные карточки и т. д.

В данной главе мы рассмотрим, какие специализированные средства для ведения оперативного учета предоставляет система 1С:Предприятие 7.7.

10.1. Работа с регистрами оперативного учета

Регистр оперативного учета — это средство для накопления сводной информации по документам. Документы, принадлежащие компоненте "Оперативный учет", при проведении формируют движения по регистрам оперативного учета. Сводная информация, накопленная в регистрах, используется для получения оперативных итогов (например, остатки и обороты товаров, текущие взаиморасчеты с клиентами и т. п.) и формирования отчетов.

10.1.1. Структура регистра

Чтобы создать новый регистр, необходимо выбрать раздел метаданных **Регистры**, нажать правую кнопку мыши и выбрать пункт **Новый Регистр**. Откроется форма настройки регистра (рис. 10.1).

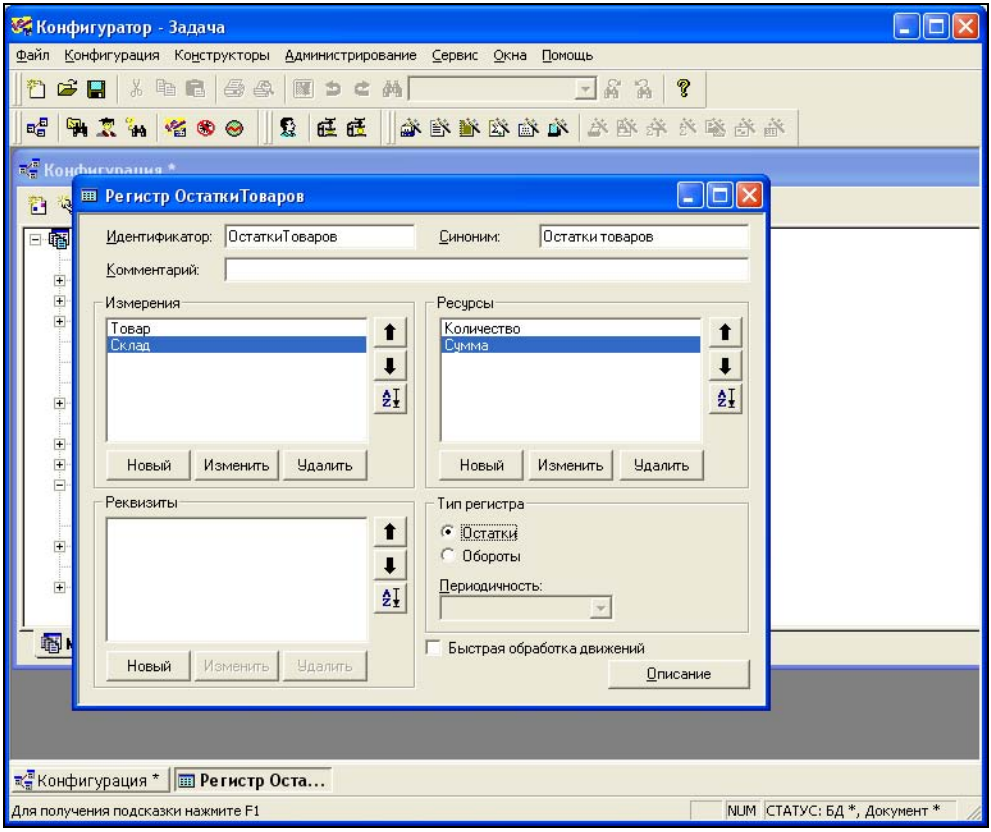


Рис. 10.1. Форма настройки регистра

В форме настройки нужно задать поля записи регистра. Поля бывают трех видов:

- **Измерения** (значения произвольного типа);
- **Ресурсы** (значения числового типа);
- **Реквизиты** (значения произвольного типа).

По измерениям информация группируется (сводится), по ресурсам — суммируется (накапливается), а реквизиты используются для получения детальной информации по движениям регистра.

Физически каждый регистр состоит из двух таблиц данных: итогов (файл rg*.dbf) и движений (файл ra*.dbf). Таблицы итогов позволяют получить итоги по регистру в определенные моменты времени, а для получения итогов на произвольный момент времени потребуется либо временный расчет, либо универсальный запрос по регистру. Таблица итогов хранит итоги по ресурсам в разрезе измерений с заданной периодичностью. Таблица движений хранит значения измерений, по которым выполняется движение; величины, на которые изменяются ресурсы; и значения дополнительных реквизитов.

Замечание

Отметим, что информация, хранящаяся в регистрах, является вторичной. Например, можно удалить все файлы r*.dbf, а затем перепровести все документы — и информация в регистрах восстановится.

Регистры бывают двух типов.

- **Остатки** — хранят информацию об остатках по совокупности измерений. Движения по регистрам остатков бывают двух типов: *приход* и *расход*;
- **Обороты** — хранят информацию об оборотах по совокупности измерений.

Задание 10.1

1. Создайте новую (пустую) конфигурацию.
2. Создайте справочники "Товары", "Склады", "Контрагенты".
3. Создайте регистр остатков ОстаткиТоваров с измерениями Товар, Склад и ресурсами Количество, Сумма.
4. Создайте оборотный регистр Продажи с измерениями Контрагент, Товар и ресурсами Количество, Сумма. Периодичность — месяц.
5. Сохраните конфигурацию и посмотрите структуру файлов регистров в файле 1cv7.dd.

10.1.2. Периодичность оперативных итогов

Оперативные итоги по регистрам хранятся с заданной периодичностью. Периодичность оборотного регистра задается в режиме конфигуратора в форме настройки регистра (рис. 10.2). Возможные значения периодичности: день, неделя, декада, месяц, квартал, год.

Периодичность регистров остатков задается в монопольном режиме 1С:Предприятия в форме управления оперативными итогами, которая вызывается из меню **Операции** пунктом **Управление оперативными итогами** (рис. 10.3). В поле **Периодичность сохранения остатков** показывается текущая периодичность, а по кнопке **Изменить** можно сменить периодичность. Возможные значения периодичности регистров остатков: пять дней, декада, пятнадцать дней, месяц.

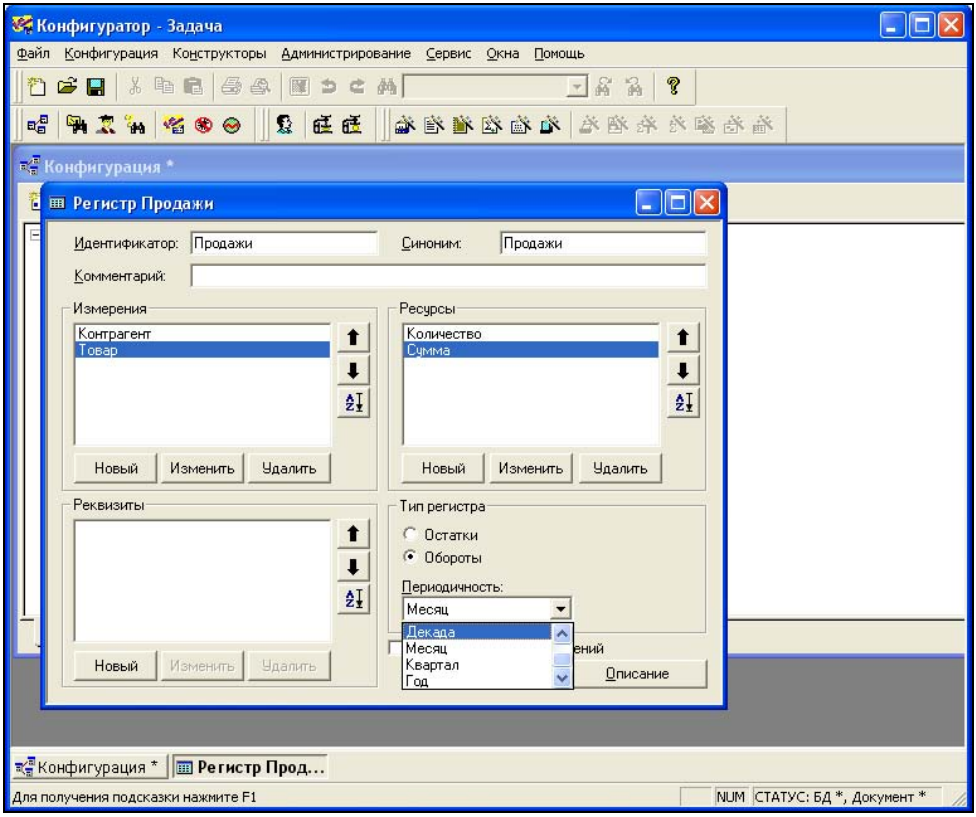


Рис. 10.2. Настройка периодичности оборотного регистра

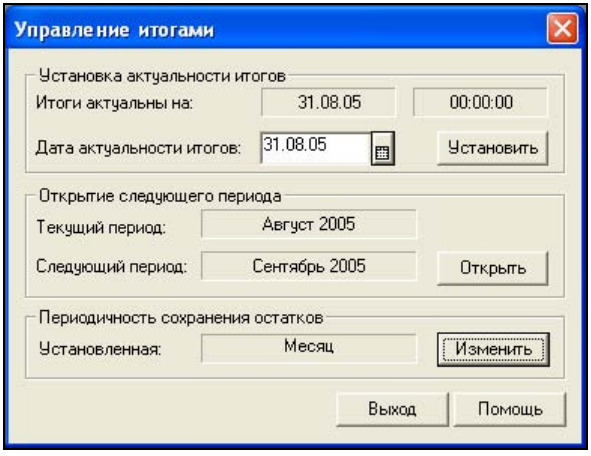


Рис. 10.3. Управление оперативными итогами

Замечание

Отметим, что в итогах регистра физически хранятся только *ненулевые остатки* по регистрам остатков и *ненулевые обороты* по оборотным регистрам.

Периодичность регистров существенно влияет на объем хранимых итогов и на время получения итогов за произвольный период. Чем меньше периодичность, тем больший объем информации хранится в таблице итогов регистра и быстрее формируются итоги за произвольный период, и, наоборот, чем больше периодичность, тем меньший объем информации хранится в таблице итогов и больше времени требуется для получения итогов за произвольный период. В типовых конфигурациях, как правило, выбирается периодичность *месяц*.

Открытие следующего периода оперативных итогов также выполняется в форме управления (рис. 10.3). Для этого в поле **Открытие следующего периода** нужно нажать кнопку **Открыть**. После этого система предложит изменить *точку актуальности* итогов и откроет форму переустановки актуальности итогов (рис. 10.4), в которой нужно отметить флажками документы (проведенные или непроведенные), которые требуется перепровести, и нажать кнопку **Выполнить**.

Чтобы открыть не один период оперативных итогов, а несколько, нужно в поле **Установка актуальности итогов** задать **Дату актуальности итогов** и нажать кнопку **Установить**, после чего также открывается форма переустановки актуальности итогов.

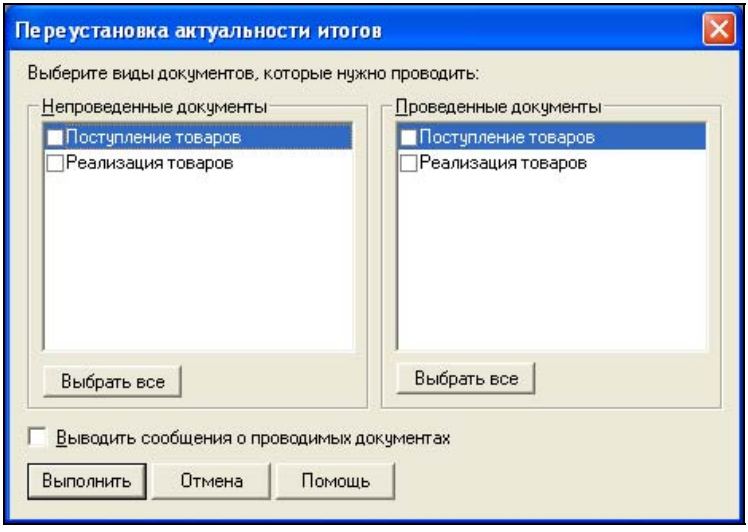


Рис. 10.4. Переустановка актуальности итогов

При открытии нового периода оперативных итогов ненулевые остатки с прошлого периода переносятся на следующий период. Отсюда следует одно очень важное правило: чтобы итоги по регистрам остатков не "разбухали", необходимо, чтобы остатки по всем ресурсам каждого из совокупности измерений рано или поздно выводились в ноль. Приведем несколько примеров.

- Если товар поступает на один склад, а реализуется с другого склада, то возникает ситуация, что остаток товара по всем складам равен нулю, а остаток этого товара по одному складу положительный, а по другому отрицательный. Такие остатки взаимно не уничтожаются, а будут переходить из периода в период.
- Следует обращать внимание на ошибки округлений. Так, например, если из-за округлений количественный остаток какого-то товара будет равен нулю, а суммовой — несколько копеек, то этот остаток будет переходить из периода в период.

Документы не могут формировать движения по будущим периодам, период должен быть предварительно открыт. Открытие периода можно выполнить и программно на встроенном языке, это будет описано далее.

10.1.3. Точка актуальности

Точка актуальности — это момент времени, на который возвращают итоги функции получения остатков по регистру остатков. Функции получения итогов по оборотному регистру по умолчанию возвращают обороты за период, которому принадлежит точка актуальности.

Управлять точкой актуальности можно только при наличии компоненты "Оперативный учет" несколькими способами:

- вручную:
  - в форме управления оперативными итогами;
  - в журнале документов, если подвести курсор на документ и нажать правую кнопку мыши, то будет доступен пункт контекстного меню **Установить ТА на документ**;
- программно — с помощью системных функций:
  - `УстановитьТАна(<ПоложениеТА>)`
  - `УстановитьТАпо(<ПоложениеТА>)`

  В этих функциях `ПоложениеТА` — это либо дата новой точки актуальности, либо позиция документа. Функция `УстановитьТАна(<ПоложениеТА>)` устанавливает точку актуальности на начало заданной даты или позиции документа, а функция `УстановитьТАпо(<ПоложениеТА>)` — на конец. Эти функции работают только в монопольном режиме (листинг 10.1);

Листинг 10.1. Установка точки актуальности на текущую дату

```
Если МонопольныйРежим()=1 Тогда
        УстановитьТАпо(ТекущаяДата());
КонецЕсли;
```

- автоматически происходит перемещение точки актуальности при проведении нового документа после точки актуальности, но в пределах текущего периода оперативных итогов. В этом случае точка актуальности устанавливается на позицию этого документа.

Если вводить документы оперативно, то есть после точки актуальности, то текущие остатки по регистрам на момент проведения документа будут актуальными.

Если вводить документы "задним числом", то есть до точки актуальности, то текущие остатки по регистрам могут не совпадать с остатками на момент проведения документа, и для получения этих остатков потребуется проводить временный расчет или выполнять запрос.

10.1.4. Запись движений по регистру

Запись движений по регистру выполняется при проведении документа, у которого в настройках стоят флажки **Разрешить проведение документа** и **Оперативный учет** (рис. 10.5).

В модуле документа в процедуре `ОбработкаПроведения()` должны быть описаны действия по записи движений по регистрам. Для доступа к регистрам есть системная константа `Регистр`. Сначала нужно присвоить значения всем измерениям, ресурсам и реквизитам регистра, а затем выполнить запись движения одной из следующих функций:

- `ДвижениеПриходВыполнить()` — для записи прихода по регистру остатков;
- `ДвижениеРасходВыполнить()` — для записи расхода по регистру остатков;
- `ДвижениеВыполнить()` — для записи движения по регистру оборотов.

Пример записи движений по регистру `ОстаткиТоваров` приведен в листинге 10.2.

Листинг 10.2. Запись прихода по регистру `ОстаткиТоваров` в модуле документа "ПоступлениеТоваров"

```
Процедура ОбработкаПроведения()
ВыбратьСтроки();
Пока ПолучитьСтроку() = 1 Цикл
```

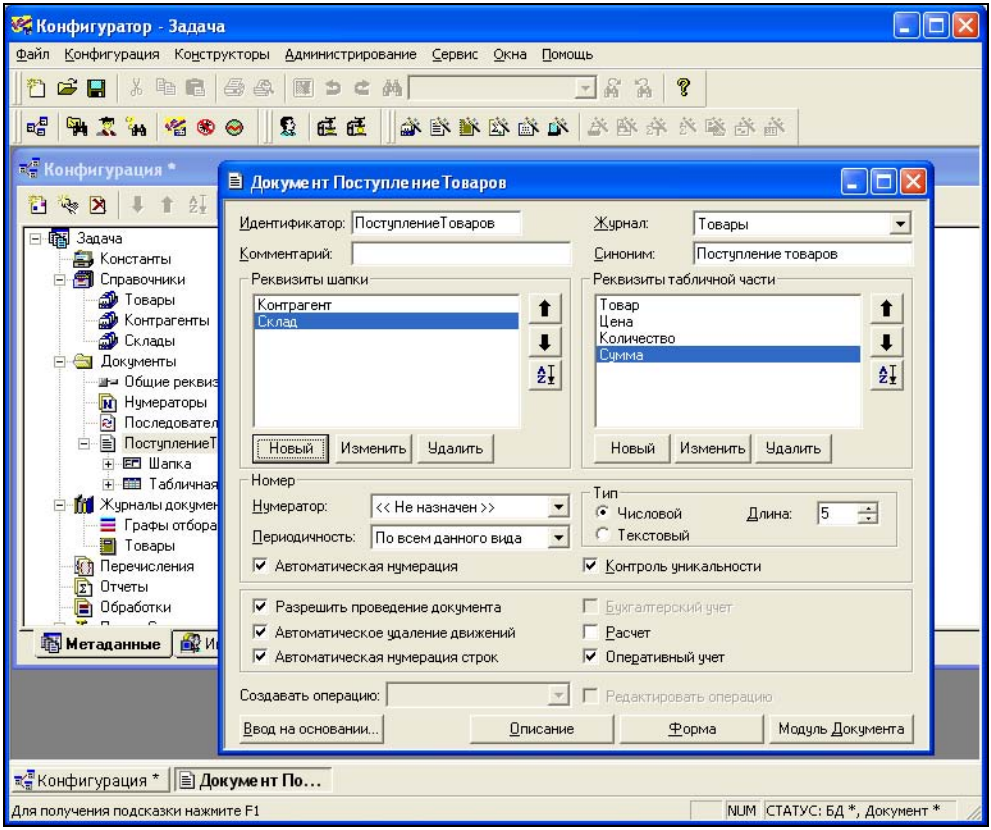


Рис. 10.5. Настройки документа

```
Регистр.ОстаткиТоваров.Товар = Товар;
        Регистр.ОстаткиТоваров.Склад = Склад;
        Регистр.ОстаткиТоваров.Количество = Количество;
        Регистр.ОстаткиТоваров.Сумма = Сумма;
        Регистр.ОстаткиТоваров.ДвижениеПриходВыполнить();
КонецЦикла;
КонецПроцедуры
```

Чтобы просмотреть, какие движения сделал документ, нужно разместить курсор на документ в журнале документов и в меню **Действия** выбрать пункт **Движения документа**. Откроется список регистров, в котором флажками отмечены те, по которым есть движения. Выбрав интересующий регистр, мы увидим окно с движениями (рис. 10.6).

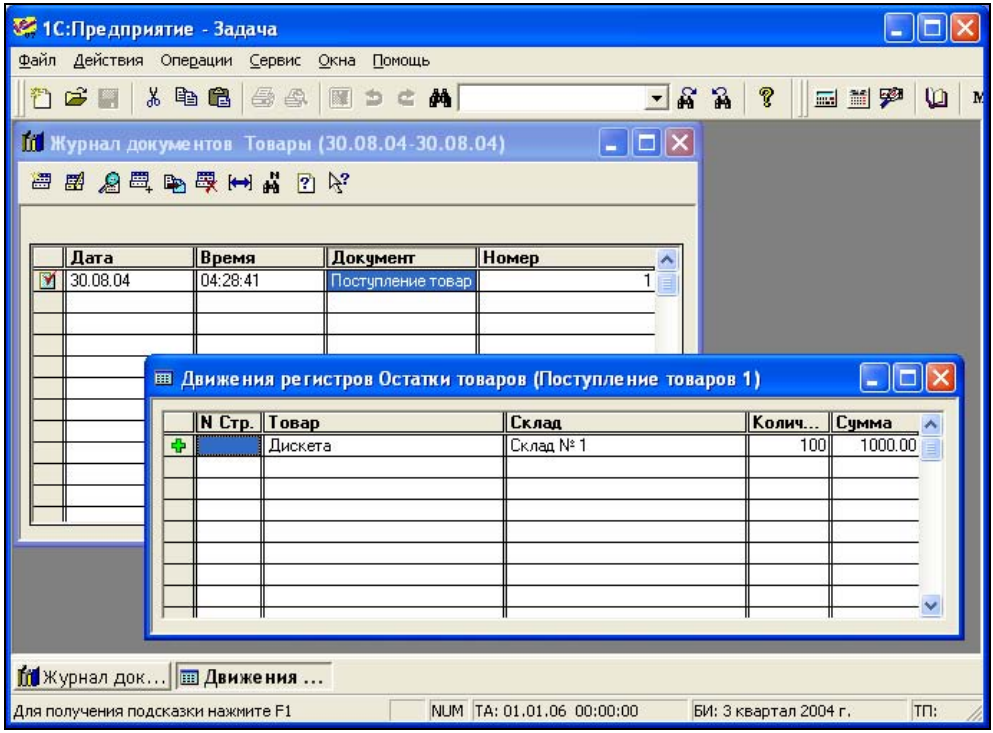


Рис. 10.6. Движения документа "Поступление товаров 1" по регистру ОстаткиТоваров

Замечание

Если, не закрывая окно движений регистра, перемещаться по списку документов, то окно движений будет автоматически обновляться, отображая движения текущего документа. Одновременно может быть открыто несколько окон с движениями по разным регистрам.

Задание 10.2

1. Создайте документ "ПоступлениеТоваров" с реквизитами шапки "Контрагент" (тип Справочник.Контрагенты), "Склад" (тип Справочник.Склады) и реквизитами табличной части "Товар" (тип Справочник.Товары), "Цена", "Количество", "Сумма" (все тип Число). При проведении документа должна выполняться запись движений прихода по регистру ОстаткиТоваров.
2. Создайте документ "Реализация" с реквизитами шапки "Контрагент" (тип Справочник.Контрагенты), "Склад" (тип Справочник.Склады) и рек-

визитами табличной части "Товар" (тип Справочник.Товары), "Цена", "Количество", "Сумма" (все тип Число). При проведении документа должна выполняться запись движений по регистру Продажи.

10.1.5. Проведение документа и точка актуальности

При проведении документа по регистрам оперативного учета система отслеживает "оперативность" работы пользователя и, по возможности, пытается не допустить случайного ввода документов "задним числом", то есть до точки актуальности.

Непреднамеренный ввод документов задним числом может произойти, например, из-за одновременной работы нескольких пользователей и несовпадения системного времени на компьютерах.

При обнаружении попытки проведения нового документа до точки актуальности и при наличии проведенных документов между точкой актуальности и позицией документа система выдает такое сообщение: "Время документа меньше точки актуальности. Существуют более поздние проведенные документы" — и предлагает изменить время документа. Если пользователь выбирает "Изменить время документа", то документ проводится оперативно, то есть после точки актуальности, и точка актуальности сдвигается на только что проведенный документ.

Если необходимо, чтобы работа пользователя выполнялась в оперативном режиме без лишних вопросов, можно установить в форме **Настройки параметров системы** на вкладке **Оперативный учет** флажки так, как показано на рис. 10.7. В этом случае документ будет записываться и проводиться после точки актуальности.

Замечание

Параметры, вводимые в форме **Настройка параметров системы**, сохраняются в файле 1cv7.cfg, причем если у пользователя задан рабочий каталог, то эти настройки будут для него индивидуальными.

Особо следует отметить ситуацию, когда делается попытка провести документ после точки актуальности, но существуют проведенные документы между точкой актуальности и позицией проводимого документа. В этом случае система выдает сообщение "Существуют более ранние проведенные документы", и документ не проводится. Такая ситуация может возникнуть, например, когда при восстановлении последовательности документов происходит ошибка, и точка актуальности остается где-то в прошлом. Чтобы документы все-таки начали проводиться, нужно передвинуть точку актуальности "в настоящее" одним из способов, описанным в *разд. 10.1.3*.

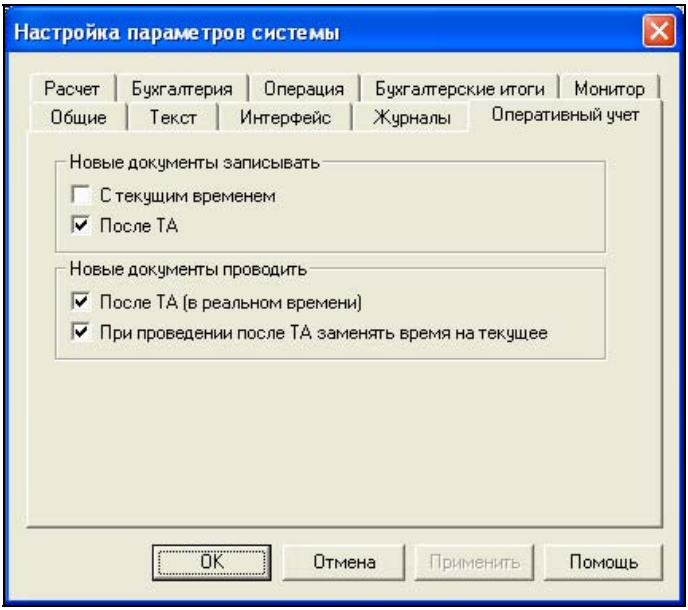


Рис. 10.7. Настройка параметров оперативного учета пользователя

Задание 10.3

Попробуйте смоделировать описанные в этом пункте ситуации.

10.1.6. Получение итогов по регистру

Получить итоги по регистру можно несколькими способами. Самый быстрый — без проведения вычислений — с помощью функций объекта `Регистр`, возвращающих итоги по совокупности измерений на *точку актуальности*.

Получение остатков

Получение остатков имеет смысл, естественно, только для регистра остатков. Функция `Остаток(<Измерен1>,<Измерен2>,...,<ИмяРесурса>)` позволяет получить текущий остаток указанного ресурса по совокупности измерений. В параметры `<Измерен1>`,`<Измерен2>`, ... передаются значения измерений, а в параметре `<ИмяРесурса>` указывается строка — идентификатор ресурса. Пример использования этой функции приведен в листинге 10.3.

Листинг 10.3. Получение количественного остатка товара на складе

```
Сообщить("Остаток товара "+Товар+ " на складе "+Склад +
" равен " + Регистр.ОстаткиТоваров.Остаток(Товар,Склад, "Количество"));
```

Чтобы узнать сколько указанного товара находится на всех складах, необходимо получить *сводный* остаток функцией `СводныйОстаток()`, которая имеет те же параметры, что и функция `Остаток()`, но некоторые значения измерений могут быть опущены (листинг 10.4).

Листинг 10.4. Получение количественного остатка товара на всех складах

```
Сообщить("Остаток товара "+Товар+ " на всех складах равен " +
Регистр.ОстаткиТоваров.СводныйОстаток(Товар, , "Количество"));
```

Если у регистра много ресурсов, то можно извлечь итоги функциями `Остатки()` или `СводныеОстатки()` и получить остатки через атрибуты регистра (листинг 10.5)

Листинг 10.5. Получение количественного и суммового остатка товара на всех складах

```
Регистр.ОстаткиТоваров.Остатки(Товар,Склад);

Сообщить("Остаток товара "+Товар + " на складе" + Склад +
Регистр.ОстаткиТоваров.Количество + " единиц на сумму " +

Регистр.ОстаткиТоваров.Сумма);
```

Получение оборотов

Получение оборотов допустимо для оборотных регистров. По умолчанию итоги выдаются за период, которому принадлежит точка актуальности. Однако можно указать и другой период в зависимости от размера периодичности оборотного регистра, с помощью функции `ИспользоватьПериод()`.

Функцией `Итог(<Измерен1>, <Измерен2>,…, <ИмяРесурса>)` можно получить оборот за период по заданной совокупности измерений `<Измерен1>`, `<Измерен2>`, … Чтобы получить сводные обороты, используется функция `СводныйИтог(<Измерен1>, <Измерен2>,…, <ИмяРесурса>)`, в которой некоторые измерения могут быть опущены. Если у оборотного регистра несколько ресурсов, то извлечь их все можно функциями `Итоги(<Измерен1>, <Измерен2>,… )` и `СводныеИтоги(<Измерен1>, <Измерен2>,…)`, а значения итогов можно получить через атрибуты регистра. Пример использования сводных итогов оборотного регистра приведен в листинге 10.6.

Листинг 10.6. Получение суммы и объема продаж товара по всем контрагентам

```
// Задаем период, за который нас интересуют продажи, – август 2005 года
Регистр.Продажи.ИспользоватьПериод(2005,8);
```

```
Регистр.Продажи.СводныеИтоги(,Товар);
Сообщить("За август 2005 года было продано " +
Регистр.Продажи.Количество + " единиц товара " + Товар +
" на сумму" + Регистр.Продажи.Сумма);
```

10.1.7. Временный расчет

Для получения итогов по регистрам не на точку актуальности, а на момент времени более ранний, чем точка актуальности, выполняется временный расчет. Сам временный расчет выполняется с помощью системных функций `РассчитатьРегистрыНа(<Период>)` и `РассчитатьРегистрыПо(<Период>)`. В качестве параметра `<Период>` указывается либо дата, либо позиция документа. Первая функция выполняет расчет на начало периода (начало дня или момент до проведения документа), а вторая — на конец периода (конец дня или момент после проведения документа).

Для получения итогов создается переменная агрегатного типа `Регистр`, и функцией `ВременныйРасчет()` включается принадлежность этого регистра к временному расчету. Затем через атрибуты и функции, описанные в *разд. 10.1.6*, производится получение итогов.

Если нас интересует итог только по некоторым значениям одного или нескольких измерений, то перед выполнением временного расчета имеет смысл установить фильтр с помощью функции `УстановитьЗначениеФильтра(<ИзмерИлиРеквизит>, <Значен>, <Вариант>)`.

Например, чтобы рассчитать среднюю себестоимость единицы товара `ВыбТовар` на конец даты `ВыбДата` (`ВыбТовар` и `ВыбДата` — реквизиты диалога формы отчета), можно воспользоваться листингом 10.7.

Листинг 10.7. Расчет себестоимости

```
Рег = СоздатьОбъект("Регистр.ОстаткиТоваров");
Рег.ВременныйРасчет();
Рег.УстановитьЗначениеФильтра("Товар",ВыбТовар);
РассчитатьРегистрыПо(ВыбДата);
Рег.СводныеОстатки(ВыбТовар,);
ОстКол = Рег.Количество;
ОстСум = Рег.Сумма;
Если ОстКол>0 Тогда
```

```
        СредняяСебестоимость=ОстСум/ОстКол;
КонецЕсли;
```

Выполнение временного расчета имеет смысл только в том случае, если на требуемый момент времени итоги не актуальны. Чтобы проверить, актуальны ли итоги в момент проведения документа, в модуле документа можно воспользоваться функцией `ИтогиАктуальны()`.

Теперь мы можем выполнить движение документа `"Реализация"` по регистру `ОстаткиТоваров`. Процедура обработки проведения этого документа приведена в листинге 10.8. Алгоритм проведения следующий: сначала мы проверяем актуальность итогов и, если итоги не актуальны, выполняем временный расчет. Перед расчетом регистров мы устанавливаем фильтр по списку товаров и складу, которые мы берем из реквизитов документа. Далее в цикле мы перебираем строки табличной части документа. Если остатка на складе по товару из текущей строки недостаточно для отгрузки, то мы отказываемся от проведения документа. Если остаток товара по складу превышает затребованное количество, то мы рассчитываем среднюю себестоимость этого товара. Наконец, если остаток совпадает с затребованным количеством, то мы списываем весь суммовой остаток, чтобы избавиться от ошибок округления, которые возникают при расчете средней цены по товару.

Листинг 10.8. Обработка проведения документа `"Реализация"`

```
Процедура ОбработкаПроведения()
Рег = СоздатьОбъект("Регистр.ОстаткиТоваров");
СписокТоваров=СоздатьОбъект("СписокЗначений");
ВыгрузитьТабличнуюЧасть(СписокТоваров,"Товар");
// Устанавливаем фильтр по вхождению значений измерения
// Товар в список значений
Рег.УстановитьЗначениеФильтра("Товар",СписокТоваров,2);
Рег.УстановитьЗначениеФильтра("Склад",Склад);
Если ИтогиАктуальны()=0 Тогда
        // Делаем временный расчет
        Рег.ВременныйРасчет();
        РассчитатьРегистрыНа(ТекущийДокумент());
КонецЕсли;
ВыбратьСтроки();
Пока ПолучитьСтроку() = 1 Цикл
    // Движение по регистру Продажи
```

```
        Регистр.Продажи.Товар = Товар;
        Регистр.Продажи.Контрагент = Контрагент;
        Регистр.Продажи.Количество = Количество;
        Регистр.Продажи.Сумма = Сумма;
        Регистр.Продажи.ДвижениеВыполнить();
        // Рассчитываем остатки по текущему товару
        Рег.Остатки(Товар,Склад);
        ОстКол = Рег.Количество;
        ОстСум = Рег.Сумма;
        Если ОстКол>Количество Тогда
                // Рассчитываем среднюю себестоимость
                Себестоимость=ОстСум/ОстКол*Количество;
        ИначеЕсли ОстКол=Количество Тогда
                // Если списываем последнюю единицу товара,
                // то, чтобы избавиться от ошибок округления,
                // за себестоимость берем всю оставшуюся сумму
                Себестоимость=ОстСум;
        Иначе
                Сообщить("В наличии "+ОстКол+" ед. товара "+Товар+
                " на складе "+ Склад+" из требуемых "+Количество);
                НеПроводитьДокумент();
                Возврат;
        КонецЕсли;
        // Движение по регистру ОстаткиТоваров
        Регистр.ОстаткиТоваров.Товар = Товар;
        Регистр.ОстаткиТоваров.Склад = Склад;
        Регистр.ОстаткиТоваров.Количество = Количество;
        Регистр.ОстаткиТоваров.Сумма = Себестоимость;
        Регистр.ОстаткиТоваров.ДвижениеРасходВыполнить();
КонецЦикла;
КонецПроцедуры
```

Задание 10.4

1. Добавьте проведение документа "Реализация" по регистру ОстаткиТоваров. Проверьте корректность проведения документа, вводя документы оперативно и "задним числом".
2. Разработайте документ "ПеремещениеТоваров", который выполняет перемещение товара с одного склада на другой, и напишите процедуру проведения этого документа по регистру ОстаткиТоваров.

10.1.8. Выполнение запросов по регистрам

Наиболее универсальным механизмом для извлечения итогов по регистрам является использование запросов *(см. главу 5)*. При выполнении запроса по регистрам имеется ряд особенностей, которые мы сейчас опишем.

Период запроса

Если в описании запроса период опущен, то интервал дат формирования запроса устанавливается в точку актуальности итогов. Если указана только начальная дата, то интервал дат будет с начальной даты по точку актуальности. Если конечная дата находится после точки актуальности, то при выполнении запроса произойдет ошибка, и будет выдано сообщение "Невозможно обращение к итогам после ТА". Эта же ошибка происходит, если точка актуальности принадлежит конечной дате запроса, так как запрос выполняется с начала дня начальной даты по конец дня конечной даты. В листинге 10.9 показано, как можно проверить, выходит ли заданный пользователем в переменных `НачДата` и `КонДата` период за точку актуальности. Если переменная `КонДата` больше либо равна дате точки актуальности, то запрос будем выполнять по точку актуальности.

Листинг 10.9. Установка периода запроса

```
ТекстЗапроса= "";
Если КонДата>=ПолучитьДатуТА() Тогда
        ТекстЗапроса = ТекстЗапроса + "Период С НачДата;";
Иначе
        ТекстЗапроса = ТекстЗапроса + "Период С НачДата По КонДата;";
КонецЕсли;
```

Путь к атрибутам регистра

Для получения значения измерения, ресурса или реквизита регистра используется определение внутренней переменной запроса, например, "`Товар = Регистр.Продажи.Товар;`". Чтобы получить ссылку на документ, совершивший движение по регистру, определяется переменная типа "`Док = Регистр.Продажи.ТекущийДокумент;`".

При работе с регистрами имеется возможность обращения к общим реквизитам документов:

```
Реквизит = Регистр.ИмяРегистра.ТекущийДокумент.ИмяОбщегоРеквизита;
```

и к самим полям документов. Но для этого необходимо указать, из каких типов документов следует выбирать значения:

```
Поле = Регистр.ИмяРегистра.ТекущийДокумент.ИмяДокумента.ИмяПоля;
```

При описании поля документа нужно обязательно указывать имя самого документа, даже если такое поле встречается в каждом документе конфигурации. Для решения этой проблемы надо либо объявить поле общим реквизитом документов, либо перечислить в описании переменной все необходимые типы документов:

```
Поле=  Регистр.ИмяРегистра.ТекущийДокумент.ИмяДокумента1.ИмяПоля,
Регистр.ИмяРегистра.ТекущийДокумент.ИмяДокументаN.ИмяПоля;
```

Использование функций в запросе по регистрам

Для ресурса регистра остатков, определенного в качестве внутренней переменной запроса, можно использовать только следующие функции:

- `НачОст(<внутр_перем>)` — вычисляет начальный остаток ресурса регистра;
- `Приход(<внутр_перем>)` — вычисляет увеличение ресурса регистра за период;
- `Расход(<внутр_перем>)` — вычисляет уменьшение ресурса регистра за период;
- `КонОст(<внутр_перем>)` — вычисляет конечный остаток ресурса регистра.

Для ресурса регистра оборотов можно использовать только функцию `Сумма(<внутр_перем>)`, которая вычисляет сумму оборота ресурса.

Внимание!

Если при описании запроса по регистрам в тексте запроса не определить функции, то результат запроса будет пустым.

Пример описания запроса, выдающего объем продаж в сумме и количестве, приведен в листинге 10.10 с учетом того, что период уже определен в переменной `ТекстЗапроса` в листинге 10.9.

Листинг 10.10. Текст запроса по продажам

```
ТекстЗапроса = ТекстЗапроса + "
|Товар = Регистр.Продажи.Товар;
|Контрагент = Регистр.Продажи.Контрагент;
|Сумма = Регистр.Продажи.Сумма;
```

```
|Количество = Регистр.Продажи.Количество;
|Группировка Товар;
|Группировка Контрагент;
|Функция ОборотСум=Сумма(Сумма);
|Функция ОборотКол=Сумма(Сумма);
";
```

Задание 10.5

1. Напишите отчет по продажам товаров с использованием запроса. В диалоге формы должны выбираться период, товар (если не выбран, то все), контрагент (если не выбран, то все). Предусмотреть группировки по товарам, контрагентам и документам движения.
2. Напишите отчет по остаткам товаров с использованием запроса. В диалоге формы должны выбираться дата, товар (если не выбран, то все), склад (если не выбран, то все). Предусмотреть группировки по товарам, складам и документам движения.

10.1.9. Регистры или бухгалтерские счета?

Между регистрами оперативного учета и бухгалтерскими счетами есть много общего (табл. 10.1). Например, и бухгалтерский счет 41.1 "Товары на складах", и регистр "Остатки товаров" накапливают информацию об остатках товаров в разрезе складов в суммовом и количественном виде. Что лучше использовать для решения конкретной учетной задачи (естественно, при наличии свободы выбора) — регистры оперативного учета или объекты бухгалтерского учета?

Ответ на этот вопрос выбирается по трем критериям:

- удобство для разработчика;
- быстродействие;
- объем базы данных.

Если разработчик — специалист по "1С" — имеет больший опыт в какой-либо из компонент, то лучше базироваться на этой компоненте. Далее, быстродействие учетной системы связано с объемом хранимой *промежуточной* информации. На регистрах можно задать периодичность хранения итогов в пять дней, а бухгалтерские итоги хранятся с фиксированной периодичностью один месяц. Таким образом, оперативные итоги при правильном проектировании структуры регистра могут формироваться быстрее, чем бухгалтерские итоги.

Таблица 10.1. *Сравнение объектов компонент "Бухгалтерский учет" и "Оперативный учет"*

Оперативный учет	Бухгалтерский учет	Комментарий
Измерения регистра. Количество измерений регистра не ограничено	*Виды субконто* на счете. Количество видов субконто на счете — не более трех в базовых и стандартных версиях, не более пяти — в профессиональных и сетевых. В качестве общего измерения для всех счетов можно рассматривать *разделитель учета*	При необходимости использовать количество измерений более пяти можно воспользоваться несколькими бухгалтерскими счетами (естественно, остальные счета должны быть забалансовыми). Именно такое решение было использовано в конфигурации "Бухгалтерия для бюджетных организаций, редакция 5" для учета материальных запасов
Ресурсы регистра. Количество ресурсов не ограничено	Ресурсов у счета не менее одного, но не более трех: `Сумма`, `Количество` (в случае ведения количественного учета на счете), `ВалСумма` (в случае ведения валютного учета на счете)	Можно ввести несколько счетов — каждый счет будет соответствовать своему ресурсу
Реквизиты регистра. Количество реквизитов не ограничено	Дополнительные реквизиты проводки. Количество реквизитов не ограничено	
Движения по регистру	Проводка между счетами (если счета балансовые) или по счету (если счет забалансовый)	
Актуальные итоги извлекаются функциями объекта "`Регистр`" и с помощью универсальных запросов	Остатки и обороты по счетам можно получить функциями объекта "`БухгалтерскиеИтоги`", а также с помощью бухгалтерского запроса	

Таблица 10.1 *(окончание)*

Оперативный учет	Бухгалтерский учет	Комментарий
Точка актуальности	Точка актуальности не поддерживается, но есть конец рассчитанного периода бухгалтерских итогов	Поскольку в бухгалтерском учете невозможно определить, актуальны бухгалтерские итоги на момент проведения документа или нет, то при проведении для получения текущих итогов *всегда выполняется либо временный расчет, либо бухгалтерский запрос*

Замечание

Если рассматривать историю развития системы 1С:Предприятие 7.7, то первоначально была выпущена компонента "Оперативный учет", в которой итоги накапливались в регистрах, а затем уже компонента "Бухгалтерский учет" с механизмом бухгалтерских итогов. В восьмой версии 1С:Предприятия выполнена некоторая унификация: вместо операций и проводок появляется объект "Регистр бухгалтерии", а регистры оперативного учета стали называться регистрами накопления. Работа с любыми регистрами (а всего их в восьмой версии четыре: сведений, накопления, бухгалтерии и расчета) выполняется с помощью механизма запросов.

10.2. Последовательности документов

Регистры оперативного учета накапливают информацию о движениях документов, причем в движениях могут храниться данные, которые в документе отсутствуют, а получаются расчетным путем. Например, в регистре `ОстаткиТоваров` есть ресурс `Сумма`, который содержит информацию о себестоимости товара на складе. При поступлении товара ресурс `Сумма` заполняется суммой из документа, а при реализации рассчитывается исходя из средней стоимости товара на момент проведения документа (см. листинг 10.8). В свою очередь, себестоимость товара зависит от того, по каким ценам и в каком количестве производились закупки товара. Если произвести корректировку документов "задним числом", то движения регистра могут стать некорректными.

Задание 10.6

1. Введите документы в следующем порядке:
   - 05.01, Поступление товаров: Дискета, 100 штук по цене 10 рублей;
   - 10.01, Поступление товаров: Дискета, 100 штук по цене 11 рублей;
   - 15.01, Реализация: Дискета, 150 штук по цене 12 рублей;

- 07.01, Реализация: Дискета, 50 штук по цене 12 рублей.

Удастся ли провести документы в такой последовательности? Будут ли корректными движения по регистрам? А итоги? Где будет находиться точка актуальности?

2. Введите дополнительно документ

- 08.01, Реализация: Дискета, 1 штука по цене 12 рублей

и ответьте на те же самые вопросы.

Конечно же, регистры оперативного учета ускоряют получение отчетов и текущих итогов, так как каждый документ при проведении делает маленький расчетик и записывает его результат в регистр. Но для того чтобы итоги по регистрам соответствовали документам, необходим механизм восстановления итогов после действий, выполненных "задним числом".

Например, описанную выше ситуацию можно решить следующим образом. Надо перепровести документы не в порядке их ввода, а в хронологическом порядке. Тогда при проведении документа от 15.01 будет выдано сообщение "В наличии 149 ед. товара `Дискета` из требуемых 150" и документ проведен не будет.

Замечание

Групповое перепроведение документов можно выполнить в монопольном режиме. Для этого нужно выбрать в меню **Операции** пункт **Проведение документов** (рис. 10.8), задать период, виды проводимых документов и нажать кнопку **Выполнить**.

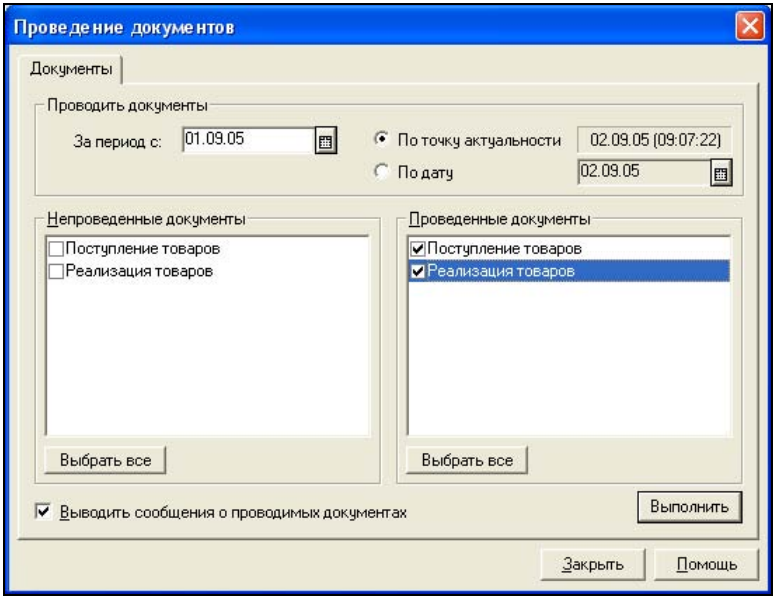


Рис. 10.8. Групповое проведение документов

Однако для перепроведения документов нужно ответить на вопросы: "С какого момента нужно перепроводить документы?" и "Какие виды документов нужно перепровести?".

Для получения ответов на эти вопросы в системе 1С:Предприятие используется механизм последовательностей документов.

10.2.1. Создание последовательности документов

Последовательности документов — это объект метаданных, который используется совместно с компонентами "Бухгалтерский учет" и "Оперативный учет" для отслеживания проведения документов в хронологическом порядке. Для создания новой последовательности нужно в разделе метаданных **Документы** встать на подраздел **Последовательности**, нажать правую кнопку мыши и выбрать пункт **Новая Последовательность Документов** (рис. 10.9).

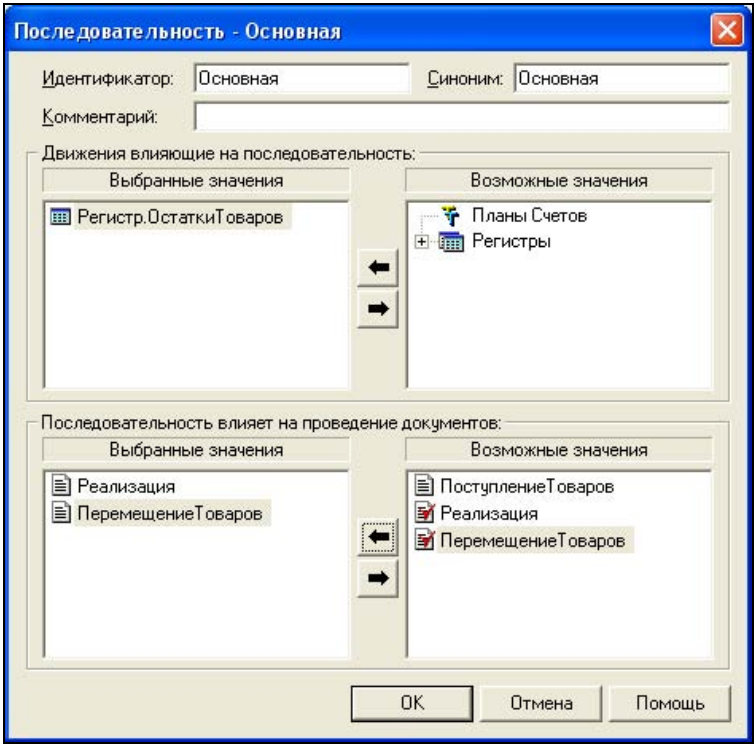


Рис. 10.9. Создание последовательности документов

В форме последовательности нужно определить, движения каких объектов влияют на последовательность (это могут быть регистры или бухгалтерские

счета) и какие виды документов нужно перепроводить при восстановлении последовательности документов.

Задание 10.7

Создайте последовательность `Основная`, как показано на рис. 10.9. Почему движения по регистру `Продажи` не влияют на последовательность? Почему документ "`ПоступлениеТоваров`" не надо перепроводить при восстановлении последовательности?

10.2.2. Граница последовательности

У последовательности есть *граница* — момент времени, с которого нарушена хронологическая последовательность проведения документов.

Чтобы узнать, где сейчас находится граница последовательности, нужно открыть форму **Проведение документов** (см. рис. 10.8) и перейти на вкладку **Последовательности** (эта вкладка появляется, если в конфигурации есть хотя бы одна последовательность, рис. 10.10). Для каждой последовательности в скобках показывается граница последовательности.

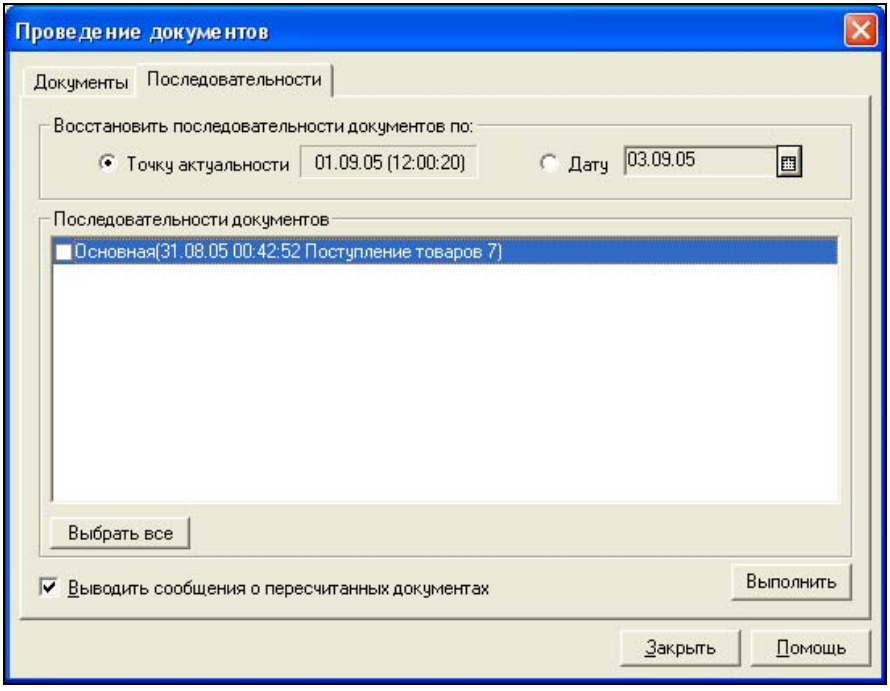


Рис. 10.10. Последовательности документов

Границу последовательности можно получить также программно на встроенном языке. Для доступа к последовательности используется системная константа `Последовательность`. Позицию границы последовательности можно получить с помощью функции `ПолучитьПозицию()`, как показано в листинге 10.11.

Листинг 10.11. Получение границы последовательности

```
Сообщить("Граница последовательности: "+
        Последовательность.Основная.Получить()+
        Последовательность.Основная.ПолучитьДокумент());
Сообщить("Позиция границы последовательности: "+
        Последовательность.Основная.ПолучитьПозицию());
```

Управление границей последовательности

Положением границы последовательности система 1С:Предприятие 7.7 управляет автоматически в процессе ввода и проведения документов. Для границы последовательности есть три стратегии поведения: "двигаться вперед", "двигаться назад" и "оставаться на месте". Для определения стратегии поведения границы последовательности необходимо сравнить текущее положение границы последовательности и позицию проводимого документа, а также проверить наличие других проведенных документов между границей последовательности и проводимым документом. Естественно, учитываться должны только те документы, движения которых влияют на границу последовательности.

Рассмотрим эти стратегии:

- *"двигаться вперед"* — позиция проводимого документа больше границы последовательности. Между документом и границей последовательности нет проведенных документов. Тогда граница последовательности устанавливается на позицию проводимого документа;
- *"оставаться на месте"* — позиция проводимого документа больше границы последовательности. Между документом и границей последовательности есть проведенные документы. Тогда граница последовательности не изменяется;
- *"двигаться назад"* — позиция проводимого документа меньше границы последовательности. Тогда граница последовательности устанавливается на позицию проводимого документа.

Таким образом, если какой-либо документ нарушает хронологическую последовательность проведения документов, то граница последовательности откатывается на него и остается там до тех пор, пока не будет выполнено последовательное перепроведение документов, начиная со следующего

документа за границей последовательности. Естественно, что перепроводиться должны не все документы, а только принадлежащие последовательности.

Границу последовательности можно также передвинуть программно без проведения документов (листинг 10.12). Однако пользоваться этой возможностью следует только в крайних случаях, например, когда в системе еще нет ни одного документа.

Листинг 10.12. Установка границы последовательности на текущую дату

```
Последовательность.Основная.Установить(ТекущаяДата());
```

10.2.3. Восстановление последовательности

Восстановить последовательность — это значит перепровести документы, принадлежащие последовательности, в хронологическом порядке. Эту операцию можно выполнить на вкладке **Последовательности** формы **Проведение документов** (см. рис. 10.10). Флажками нужно отметить восстанавливаемые последовательности, указать конечную дату или точку актуальности и нажать кнопку **Выполнить**.

Восстановить последовательность можно также программно на встроенном языке. Для этого нужно выбрать документы, принадлежащие последовательности, и перепровести их в цикле, как показано в листинге 10.13. Приведенная процедура восстановления последовательности будет выполняться и не в монопольном режиме. Однако она отличается от системной процедуры тем, что система при восстановлении последовательности одновременно передвигает точку актуальности, чтобы на момент проведения документа итоги были актуальны.

Листинг 10.13. Восстановление последовательности

```
Послед=Последовательность.Основная;
Док=СоздатьОбъект("Документ");
Док.ВыбратьДокументы(Послед.ПолучитьПозицию(),);
Пока (Док.ПолучитьДокумент()>0) Цикл
        Если Док.Проведен()=0 Тогда  Продолжить; КонецЕсли;
        ТекДок = Документ.ТекущийДокумент();
        Если Послед.ПринадлежитПоследовательности(ТекДок)>0 Тогда
                Объект=СоздатьОбъект("Документ");
                Объект.НайтиДокумент(ТекДок);
```

```
            Попытка
                Если Объект.Провести() = 0 Тогда
                    Прервать;
                КонецЕсли;
            Исключение
                Сообщить("Не удалось провести документ
                "+Строка(Объект)+" Ошибка: "+ОписаниеОшибки());
                Прервать;
            КонецПопытки;
        КонецЕсли;
КонецЦикла;
```

Задание 10.8

1. Создайте обработку "Управление последовательностями", которая позволяет переместить границу последовательности на дату, заданную в реквизите диалога, без перепроведения документов (см. листинг 10.12) и с перепроведением документов (см. листинг 10.13).
2. Смоделируйте ситуации, при которых будут наблюдаться все стратегии поведения границы последовательности при проведении документов.
3. Сделайте задание, приведенное в табл. 9.2 *(см. разд. 9.2.1)*, с использованием механизма последовательностей.

10.3. Пример решения экзаменационной задачи

Особенности проведения аттестационного экзамена по компоненте "Оперативный учет" приведены на сайте **www.1c.ru**. Отметим основные моменты.

10.3.1. Условия проведения экзамена

Условия проведения экзамена следующие.

- Экзамен проходит в форме решения практической задачи — разработки "с нуля" простой конфигурации по постановке, изложенной в задании. На решение отводится 4 часа. Аттестуемый должен предложить проект конфигурации — перечень и структуру объектов метаданных — и создать эти объекты в пустой конфигурации, в том числе отладить программные модули объектов, разработать указанные в задании печатные формы.
- Особое внимание при решении задачи должно быть уделено проектированию специфических для компоненты "Оперативный учет" объектов

метаданных — регистров. Решения, в которых учет построен не на регистрах, к защите не принимаются.

- В тех случаях, когда в задании явно указаны структуры объектов, типы и свойства элементов данных и методы встроенного языка, с помощью которых должна быть построена логика системы учета, аттестуемый обязан их использовать. В противном случае автор конфигурации вправе самостоятельно принимать проектные решения.

10.3.2. За что снижается оценка

Оценка снижается за ошибки при проектировании структур регистров и ошибки при разработке алгоритмов обработки данных регистров. Вот примеры грубых ошибок:

- нельзя на регистрах остатков вести учет ресурсов, принципиально не выводимых в ноль. Плохо, когда ресурсы регистра остатков (один или все) изменяются документами только "в одну сторону" (только в "плюс" или только в "минус"), то есть не обеспечивается выведения остатков ресурсов в ноль;
- нельзя допускать рассогласование по набору измерений при выполнении положительных и отрицательных движений для регистра остатков. Плохо, когда ресурсы регистра остатков (один или все) изменяются документами и в "плюс", и в "минус", но движения с противоположным знаком для одного и того же объекта учета выполняются с разными наборами значений измерений, что также не обеспечивает выведения остатков ресурсов в ноль;
- если при проведении документа используются каким-то образом остатки или итоги, считываемые из регистров, обязательно требуется предусмотреть временный расчет таких регистров в случае проведения документов задним числом на момент проведения документа;
- нельзя при проведении документов пользоваться открытием модальных окон (командой `Предупреждение()`). Поскольку проведение документа является транзакцией, то при появлении модального окна работа всей системы будет существенно приторможена, пока не закроется модальное окно;
- временный расчет всегда заключается в условии его необходимости (со строгим неравенством). Любая попытка получить временный расчет на момент, хоть на секунду (или позицию документа), стоящий после точки актуальности, приводит к предупреждению об ошибке;
- конфигурация должна устойчиво работать не только при движении вперед, но и назад. То есть, при отмене проведения любого документа состояние показателей, контролируемых системой, должно возвращаться в

исходное положение (как было до проведения документа). Тогда можно будет фактически размотать всю цепочку документов назад. Ошибкой является ситуация, когда в созданной конфигурации такое сделать невозможно;

- конфигурация должна устойчиво работать и при создании документов "задним числом";
- должны быть использованы механизмы "Последовательностей" для контроля положения точки их нарушения;
- при групповом перепроведении документов (восстановлении последовательностей) система должна четко и точно (локализованно) предупреждать пользователя о проблемах (невозможности проведения тех или иных документов), а по возможности — выдавать рекомендации по их исправлению;
- следует избегать ситуации, когда при проведении документа учитывается нечто, кроме данных самого документа или данных, взятых из регистров на момент проведения документов. Если при проведении документа учитывается состояние какого-нибудь реквизита справочника, то есть опасность, что при перепроведении документа значение реквизита будет уже совсем другое. Ошибкой является не отработка подобных действий пользователя (нет программных действий, каким-либо образом восстанавливающих логику при таких действиях или препятствующих "безответственному" применению пользователем подобных действий);
- следует избегать ситуации, когда при проведении изменяются другие документы или реквизиты справочников (исключение — периодические реквизиты методом `УстановитьРеквизитСправчника()`) или заводятся новые элементы в справочниках. Ошибкой является неотработка ситуаций: "Отмена проведения" ("Удаление документа"), "Изменение даты документа", "Изменение времени документа";
- ошибкой при работе с документами является неприменение механизмов оптимизации получения сводных итогов, временных расчетов и выборок из регистров на основе фильтров (условий) и отборов, либо ложное (неэффективное) их использование. Под ложным (неэффективным) использованием следует считать следующие случаи:
  - организация временных расчетов внутри цикла перебора строк;
  - применение фильтров (условий) по вторым и далее измерениям без отборов остатков и движений (при отсутствии флажков в структуре регистров) в случаях, когда отсутствуют фильтры по предыдущим измерениям;
  - применение методов получения сводных итогов в интерактивных режимах (например, в вычисляемых полях формы) по значениям непервых измерений, опуская предыдущие;

- применение последовательных множественных временных расчетов одних и тех же регистров с уничтожением переменных типа "регистр" или отключением/включением принадлежности их к временному расчету.

10.3.3. Задача "Автоматизация учета заявок и оплат"

Текст задания "Автоматизация учета заявок и оплат" приведен в табл. 10.2.

Таблица 10.2. *Текст задания с комментариями*

№	Текст задания	Комментарий
1	Необходимо разработать конфигурацию, которая позволяет автоматизировать учет потока заказов товаров у поставщиков и потока оплат за них. По договору с поставщиками автоматизируемая организация выставляет поставщикам "Заявки на поставку". Каждая заявка содержит информацию, в какие сроки необходимо расплатиться с поставщиком. Причем, не дожидаясь оплаты, может быть выставлена новая "Заявка на поставку", а потом произойти единая оплата (покрывающая обе заявки). Также возможны случаи предоплаты и частичных оплат. Конфигурация должна показывать состояние оплат в разрезе поставщиков, заявок и сроков их оплат. Отдельный интерес (для начисления пени) вызывают ситуации просроченных оплат и недопоставок со стороны поставщика	В данной задаче заявка выступает как соглашение, по которому поставщик берет на себя обязательство поставить товар, а покупатель (это мы) берет обязательство оплатить его в течение определенного срока. Недостающая информация в данной постановке — срок поставки товара. Можно предложить два варианта доопределения задачи. Первый — добавить в заявку реквизит "Срок поставки" (так, например, сделано в типовой конфигурации "Торговля и Склад"). Второй — считать, что срок поставки и срок оплаты совпадают. Согласно требованиям к экзамену, мы принимаем решение на свое усмотрение и выбираем второй вариант, как более простой для реализации. В пользу этого варианта говорит также то, что в случае одновременного отсутствия платежа и поставки по заявке общая сумма пени будет нулевой
2	Справочник "Поставщики" содержит реквизит "Процент пени"	Будем считать, что процент пени действует как на просрочку оплаты, так и на недопоставку
3	Регистр(ры) обеспечивает(ют) ведение учета сумм долга в разрезе поставщиков и кредитных документов	Регистр может быть один, или несколько, на усмотрение разработчика. Обязательные измерения: поставщики и кредитные документы; обязательный ресурс — долг

Таблица 10.2 *(продолжение)*

№	Текст задания	Комментарий
4	Документ "Заявка на поставку" содержит информацию о поставщике и сроке отсрочки (количество дней) платежа, а в табличной части: информацию о товарах, количестве, цене и сумме за товар (равной произведению цены и количества). При проведении документа "Заявка на поставку" необходимо проконтролировать, не было ли по данному поставщику предоплат. Если были, то документ гасит все предоплатные документы (начиная с самых старых), на сколько хватит его суммы. Если непогашенных предоплат нет, или осталась еще некая сумма после их погашения, "Заявка на поставку" *формирует кредитный долг по данному поставщику*	Погашение оплат, сделанных авансом (предоплат) выполняется по алгоритму FIFO — более ранняя предоплата погашается первой. Как мы увидим далее, погашаться должны также и начисленные пени
5	Документ "Приходная накладная" вводится на основании "Заявки на поставку". Он должен копировать всю информацию из исходной заявки. Впоследствии в документе может быть исправлено количество (автоматически должна пересчитаться сумма) или удалены строки по недошедшим товарам. Остальные реквизиты исправлениям не подлежат. Проведение такого документа означает факт полного или частичного исполнения своих обязательств со стороны поставщика (по исходной заявке)	Документ "Приходная накладная" погашает обязательство поставщика по заявке. В задании описывается только вариант жесткой связки "Заявка-Поставка", то есть приходная накладная погашает только ту заявку, которая является основанием документа, причем одну заявку может погашать несколько приходных накладных. А как быть в ситуации, когда в общей сложности поставлено товара больше, чем указано в заявке? Можно предложить такой вариант: поставка должна гасить в первую очередь заявку-основание, а затем другие заявки в порядке их сроков поставки (оплаты). Если и после этого осталось нераспределенное количество, то его нужно рассматривать как поставку в счет будущих заявок. Другой вариант — запретить приход большего количества, чем указано в заявке. Так как этот вариант проще, то его мы и реализуем

Таблица 10.2 *(окончание)*

№	Текст задания	Комментарий
6	Документ "Оплата" содержит информацию о поставщике, которому платим, и сумме оплаты. Проведение такого документа означает факт полного или частичного исполнения своих обязательств перед поставщиком (по исходной заявке). При проведении документа "Оплата" контролируется, нет ли задолженностей по данному поставщику. Если есть, то происходит их погашение (начиная с самых старых долгов), оставшаяся сумма формирует предоплату по данному поставщику	Погашение заявок выполняется по алгоритму FIFO: более ранняя заявка по сроку платежа погашается первой. Как мы увидим далее, погашаться должны также и начисленные пени
7	Регламентный документ "Начисление пени" формируется в конце дня. Назначение — начисление пени по просроченным долгам. Пени начисляются согласно "процента пени" каждого поставщика. При проведении документов "Оплата" пени гасятся на тех же принципах, как и "Заявки" (в порядке общей очереди)	Пеня рассчитывается на просроченные оплаты и на недопоставки
8	В конфигурации должен быть отчет "График платежей". В отчет должна попадать информация по задолженностям перед поставщиками. Причем информация должна обобщаться по дням, до которых необходимо сделать оплату, далее по поставщикам и заявкам	
9	В конфигурации должен быть отчет "Взаиморасчеты", показывающий в режиме "Кратко" состояние взаиморасчетов по поставщикам в разрезе кредитных документов. В режиме "Подробно" — еще и со ссылкой	

План решения

Решение любой задачи можно разбить на этапы. Мы предлагаем следующий порядок действий. Сначала нужно придумать несколько примеров с цифрами по условию задачи, которые, по возможности, показывали бы все возможные ситуации. Затем нужно описать, какие объекты используются в

задаче, какова их структура и как они взаимодействуют. В тексте задачи уже определены справочники и документы, описана их структура. Неясным является только количество и структура регистров. Затем нужно описать алгоритмы проведения документов по регистрам. Словесно они уже описаны в тексте задания, но нужно детализировать их с учетом структуры разработанных регистров. И в последнюю очередь, разрабатываем отчеты.

Для того чтобы отчеты формировались максимально быстро, они должны соответствовать структуре регистров, в противном случае для их получения потребуются дополнительные вычисления.

Тестовый пример

К разработке тестового примера нужно отнестись очень серьезно. Все равно вам придется каким-то образом проверять правильность работы программы. Однако если тесты будут разработаны *до* разработки конфигурации, то вы лучше прочувствуете задачу, и при разработке структуры данных и модулей учтете возможные ситуации.

В задаче у нас есть четыре вида документов: "Заявка на поставку", "Оплата", "Приходная накладная" и "Начисление пени". Из текста задания следует, что могут возникнуть следующие ситуации:

- А. Сначала заявка, потом оплата (кредитный документ — заявка).
- Б. Сначала оплата, потом заявка (кредитный документ — оплата).
- В. Оплата в срок (дата оплаты меньше даты заявки плюс срок оплаты).
- Г. Просроченная оплата (дата оплаты превышает дату заявки плюс срок оплаты).
- Д. Полная поставка (пришло столько товара, сколько указано в заявке).
- Е. Недопоставка (пришло товара меньше, чем в заявке).

Будем считать, что во всех примерах документов участвует один и тот же поставщик и один и тот же товар. Получить тестовые примеры с разными поставщиками и товарами можно простым копированием документов. Процент пени зададим 0,1% от суммы долга за каждый день просрочки платежа или поставки. Список тестовых примеров приведен в табл. 10.3.

Таблица 10.3. *Текст задания с комментариями*

Дата	Документ	Реквизиты документа	Ситуация	Комментарий
01.09	Заявка № 1	Количество 100, сумма 1000 р. Срок оплаты 10 дней	А	Появляется долг на сумму 1000 руб. Срок оплаты 11.09
02.09	Заявка № 2	Количество 100, сумма 1000 р. Срок оплаты 8 дней	А	Появляется долг на сумму 1000 руб. Срок оплаты 10.09

Таблица 10.3 *(продолжение)*

Дата	Документ	Реквизиты документа	Ситуация	Комментарий
10.09	Приходная накладная № 1	На основании заявки № 1 Количество 90, сумма 900	Е	Частично погашает заявку № 1
11.09	Приходная накладная № 2	На основании заявки № 2 Количество 80, сумма 800	Е	Частично погашает заявку № 2
11.09	Начисление пени № 1		Г, Е	По заявке № 2 начисляется пеня поставщику 200 × 0,1% в сумме 0,2 руб. По заявке № 2 начисляется пеня покупателю 1000 × 0,1% в сумме 1 руб. Общая сумма пени покупателя 0,8 руб.
12.09	Оплата № 1	Сумма 1500 руб.	А	Полностью погашает заявку № 2 (1000 руб.) и частично погашает заявку № 1 (500 руб.)
12.09	Начисление пени № 2		Г, Е	По заявке №1 начисляется пеня поставщику 100 × 0,1% в сумме 0,1 руб. По заявке № 2 начисляется пеня поставщику 200 × 0,1% в сумме 0,2 руб. По заявке № 1 начисляется пеня покупателю 500 × 0,1% в сумме 0,5 руб. Общая сумма пени покупателя 0,2 руб.
13.09	Оплата № 2	Сумма 1000 руб.	А, Б	Полностью погашает заявку № 1 (500 руб.), начисление пени № 1 (0,8 руб.), начисление пени № 2 (0,2 руб.) и появляется предоплата на сумму 499 руб.

Таблица 10.3 *(окончание)*

Дата	Документ	Реквизиты документа	Ситуация	Комментарий
13.09	Приходная накладная № 1	На основании заявки № 1 Количество 10, сумма 100 руб.	Д	Полностью погашает заявку № 1
13.09	Начисление пени № 3		Е	По заявке № 2 начисляется пеня поставщику 200 × 0,1% в сумме 0,2 руб.

Проектируем регистры

В нашей задаче необходимо контролировать состояние задолженности перед поставщиком. Для этого разработаем структуру регистра **Взаиморасчеты**.

Измерения

Так как взаиморасчеты ведутся в разрезе поставщиков, то определим первое измерение регистра как `Поставщики` (тип `Справочник.Поставщики`).

Наша задолженность перед поставщиком увеличивается документами `"Заявка на поставку"`, `"Начисление пени"` (за задержку платежа), уменьшается документами `"Оплата"`, `"Начисление пени"` (за задержку поставки).

В то же время задолженность поставщика перед нами увеличивается документом `"Заявка на поставку"`, а уменьшается документом `"Приходная накладная"`.

Поскольку учет оплат и поставок по заявкам производится независимо, то определим второе измерение регистра как `ВидДолга` (тип `Перечисление.ВидыДолга`). Соответствующее перечисление будет иметь два значения: `Оплата` и `Поставка`.

Взаиморасчеты должны вестись в разрезе кредитных документов, которыми могут быть как заявки, так и оплаты. Поэтому определим измерение `КредДокумент` (тип `Документ`).

Наконец, учет оплат нужно вести в разрезе сроков платежей, поэтому зададим еще одно измерение `Срок` (тип `Дата`).

Таким образом, регистр **Взаиморасчеты** будет иметь четыре измерения:

- `Поставщики`;
- `ВидДолга`;
- `КредДокумент`;
- `Срок`.

Ресурсы

Ресурс у регистра **Взаиморасчеты** будет один — `Сумма` (тип `Число`).

Задание 10.9

Разработайте структуру регистра **Заявки**, который будет показывать состояние заявки. Итоги по этому регистру должны показывать, какие товары еще должны поступить по заявкам на поставку.

Алгоритмы проведения документов

Основной алгоритм, который применяется при проведении документов `"ЗаявкаНаПоставку"` и `"Оплата"` — это алгоритм FIFO, согласно которому заявки погашают сделанные ранее предоплаты в порядке возрастания дат, а оплаты погашают заявки в порядке возрастания сроков платежей.

Реализация алгоритма заключается в следующем. Сначала мы проверяем актуальность итогов и, при необходимости, делаем временный расчет. Далее выгружаем итоги в таблицу значений и упорядочиваем ее по кредитным документам (по срокам платежей). Затем перебираем в цикле кредитные документы и погашаем их, пока не исчерпается сумма заявки (сумма оплаты).

Если после цикла сумма заявки (сумма оплаты) больше нуля, то формируем положительное движение для заявки и отрицательное — для оплаты. В этом случае кредитным документом является текущий (заявка или оплата).

Текст модулей документов приведен в листингах 10.14—10.17.

Листинг 10.14. Модуль документа `"ЗаявкаНаПоставку"`

```
Процедура ОбработкаПроведения()
СуммаЗаявки=Итог("Сумма"); // Сумма по колонке табличной части
Рег=СоздатьОбъект("Регистр.Взаиморасчеты");
Рег.УстановитьЗначениеФильтра("Поставщик",Поставщик);
// Зафиксируем обязательство поставщика по поставке товара
Регистр.Взаиморасчеты.Поставщик=Поставщик;
Регистр.Взаиморасчеты.ВидДолга=Перечисление.ВидыДолга.Поставка;
Регистр.Взаиморасчеты.КредДокумент=ТекущийДокумент();;
Регистр.Взаиморасчеты.Срок=ДатаДок+Срок;
Регистр.Взаиморасчеты.Сумма=СуммаЗаявки;
Регистр.Взаиморасчеты.ДвижениеРасходВыполнить();
// Разбросаем сумму заявки по предоплатам (если таковые есть)
Рег.УстановитьЗначениеФильтра("ВидДолга",Перечисление.ВидыДолга.Оплата);
Если ИтогиАктуальны()=0 Тогда
```

```
        // Делаем временный расчет
        Рег.ВременныйРасчет();
        РассчитатьРегистрыНа(ТекущийДокумент());
КонецЕсли;
Ит=СоздатьОбъект("ТаблицаЗначений");
Рег.ВыгрузитьИтоги(Ит);
Ит.Сортировать("КредДокумент+");
Ит.ВыбратьСтроки();
Пока Ит.ПолучитьСтроку()=1 Цикл
        Если Ит.Сумма<0 Тогда
                // Погашаем оплату
                Регистр.Взаиморасчеты.Поставщик=Поставщик;

        Регистр.Взаиморасчеты.ВидДолга=Перечисление.ВидыДолга.Оплата;
                Регистр.Взаиморасчеты.КредДокумент=Ит.КредДокумент;
                Регистр.Взаиморасчеты.Срок=Ит.Срок;
                Регистр.Взаиморасчеты.Сумма=Мин(-Ит.Сумма,СуммаЗаявки);
                Регистр.Взаиморасчеты.ДвижениеПриходВыполнить();
                СуммаЗаявки=СуммаЗаявки-Регистр.Взаиморасчеты.Сумма;
                Если СуммаЗаявки=0 Тогда
                        Прервать;
                КонецЕсли;
        КонецЕсли;
КонецЦикла;
// Оставшуюся сумму заявки запишем, как долг поставщику
Если СуммаЗаявки>0 Тогда
        Регистр.Взаиморасчеты.Поставщик=Поставщик;
        Регистр.Взаиморасчеты.ВидДолга=Перечисление.ВидыДолга.Оплата;
        Регистр.Взаиморасчеты.КредДокумент=ТекущийДокумент();;
        Регистр.Взаиморасчеты.Срок=ДатаДок+Срок;
        Регистр.Взаиморасчеты.Сумма=СуммаЗаявки;
        Регистр.Взаиморасчеты.ДвижениеПриходВыполнить();
КонецЕсли;
КонецПроцедуры
```

Листинг 10.15. Модуль документа "ПриходнаяНакладная"

```
Процедура ОбработкаПроведения()
СуммаПрихода=Итог("Сумма");
```

```
Рег=СоздатьОбъект("Регистр.Взаиморасчеты");
Рег.УстановитьЗначениеФильтра("Поставщик",Поставщик);
Рег.УстановитьЗначениеФильтра("ВидДолга",
Перечисление.ВидыДолга.Поставка);
Рег.УстановитьЗначениеФильтра("КредДокумент",Заявка);
// Проверим сумму задолженности поставщика по заявке
Если ИтогиАктуальны()=0 Тогда
        // Делаем временный расчет
        Рег.ВременныйРасчет();
        РассчитатьРегистрыНа(ТекущийДокумент());
КонецЕсли;
Срок=Заявка.ДатаДок+Заявка.Срок;
ДолгПоЗаявке=-Рег.Остаток(Поставщик,Перечисление.ВидыДолга.Поставка,
Заявка,Срок,"Сумма");
Если ДолгПоЗаявке>=СуммаПрихода Тогда
        // Обязательство поставщика по поставке товара выполнено
        (полностью или частично)
        Регистр.Взаиморасчеты.Поставщик=Поставщик;
        Регистр.Взаиморасчеты.ВидДолга=Перечисление.ВидыДолга.Поставка;
        Регистр.Взаиморасчеты.КредДокумент=Заявка;
        Регистр.Взаиморасчеты.Срок=Срок;
        Регистр.Взаиморасчеты.Сумма=СуммаПрихода;
        Регистр.Взаиморасчеты.ДвижениеПриходВыполнить();
Иначе
        Сообщить("По заявке "+Заявка+" осталось поставить товара на сумму
        "+ДолгПоЗаявке+", что превышает сумму поставки");
        НеПроводитьДокумент();
КонецЕсли;
КонецПроцедуры
```

Листинг 10.16. Модуль документа "Оплата"

```
Процедура ОбработкаПроведения()
Рег=СоздатьОбъект("Регистр.Взаиморасчеты");
Рег.УстановитьЗначениеФильтра("Поставщик",Поставщик);
// Разбросаем сумму оплаты по заявкам (если таковые есть)
Рег.УстановитьЗначениеФильтра("ВидДолга",Перечисление.ВидыДолга.Оплата);
Если ИтогиАктуальны()=0 Тогда
        // Делаем временный расчет
        Рег.ВременныйРасчет();
```

```
        РассчитатьРегистрыНа(ТекущийДокумент());
КонецЕсли;
Ит=СоздатьОбъект("ТаблицаЗначений");
Рег.ВыгрузитьИтоги(Ит);
Ит.Сортировать("Срок+");
СуммаОплаты=Сумма;
Ит.ВыбратьСтроки();
Пока Ит.ПолучитьСтроку()=1 Цикл
        Если Ит.Сумма>0 Тогда
                // Погашаем кредитный документ
                Регистр.Взаиморасчеты.Поставщик=Поставщик;
                Регистр.Взаиморасчеты.ВидДолга=
                        Перечисление.ВидыДолга.Оплата;
                Регистр.Взаиморасчеты.КредДокумент=Ит.КредДокумент;
                Регистр.Взаиморасчеты.Срок=Ит.Срок;
                Регистр.Взаиморасчеты.Сумма=Мин(Ит.Сумма,СуммаОплаты);
                Регистр.Взаиморасчеты.ДвижениеРасходВыполнить();
                СуммаОплаты=СуммаОплаты-Регистр.Взаиморасчеты.Сумма;
                Если СуммаОплаты=0 Тогда
                        Прервать;
                КонецЕсли;
        КонецЕсли;
КонецЦикла;
// Оставшуюся сумму заявки запишем, как предоплату
Если СуммаОплаты>0 Тогда
        Регистр.Взаиморасчеты.Поставщик=Поставщик;
        Регистр.Взаиморасчеты.ВидДолга=Перечисление.ВидыДолга.Оплата;
        Регистр.Взаиморасчеты.КредДокумент=ТекущийДокумент();;
        Регистр.Взаиморасчеты.Сумма=СуммаОплаты;
        Регистр.Взаиморасчеты.Срок=ДатаДок;
        Регистр.Взаиморасчеты.ДвижениеРасходВыполнить();
КонецЕсли;
КонецПроцедуры
```

Листинг 10.17. Модуль документа "НачислениеПени"

```
Процедура ОбработкаПроведения()
Рег=СоздатьОбъект("Регистр.Взаиморасчеты");
Рег.УстановитьЗначениеФильтра("Поставщик",Поставщик);
```

```
// Определим просроченные оплаты и недопоставки
Если ИтогиАктуальны()=0 Тогда
        // Делаем временный расчет
        Рег.ВременныйРасчет();
        РассчитатьРегистрыНа(ТекущийДокумент());
КонецЕсли;
Ит=СоздатьОбъект("ТаблицаЗначений");
Рег.ВыгрузитьИтоги(Ит);
Ит.Сортировать("КредДокумент+");
Ит.ВыбратьСтроки();
БазаПени=0;
Пока Ит.ПолучитьСтроку()=1 Цикл
        Если Ит.КредДокумент.Вид()="ЗаявкаНаПоставку" Тогда
                Если Ит.Срок<ДатаДок Тогда
                        БазаПени=БазаПени+Ит.Сумма;
                КонецЕсли;
        КонецЕсли;
КонецЦикла;
Если БазаПени<>0 Тогда
        СуммаПени=БазаПени*Поставщик.ПроцентПени/100;
        Регистр.Взаиморасчеты.Поставщик=Поставщик;
        Регистр.Взаиморасчеты.ВидДолга=Перечисление.ВидыДолга.Оплата;
        Регистр.Взаиморасчеты.КредДокумент=ТекущийДокумент();
        Регистр.Взаиморасчеты.Срок=ДатаДок;
        Если СуммаПени>0 Тогда
                // Увеличиваем долг поставщику
                Регистр.Взаиморасчеты.Сумма=СуммаПени;
                Регистр.Взаиморасчеты.ДвижениеПриходВыполнить();
        Иначе
                // Уменьшаем долг поставщику
                Регистр.Взаиморасчеты.Сумма=-СуммаПени;
                Регистр.Взаиморасчеты.ДвижениеРасходВыполнить();
        КонецЕсли;
КонецЕсли;
КонецПроцедуры
```

Задание 10.10

1. Добавьте в модули документов "ЗаявкаНаПоставку" и "ПриходнаяНакладная" формирование движений по регистру "Заявки".

2. Разработайте ввод документа "ПриходнаяНакладная" на основании документа "ЗаявкаНаПоставку". При заполнении табличной части должны анализироваться итоги по регистру "Заявки".

Разрабатываем последовательность

Все рассмотренные документы тесно связаны между собой. Изменение, удаление или ввод задним числом какого-либо из документов конфигурации потребует перепроведения последующих документов. Поэтому для контроля правильной последовательности проведения документов разработаем последовательность "Основная", как показано на рис. 10.11.

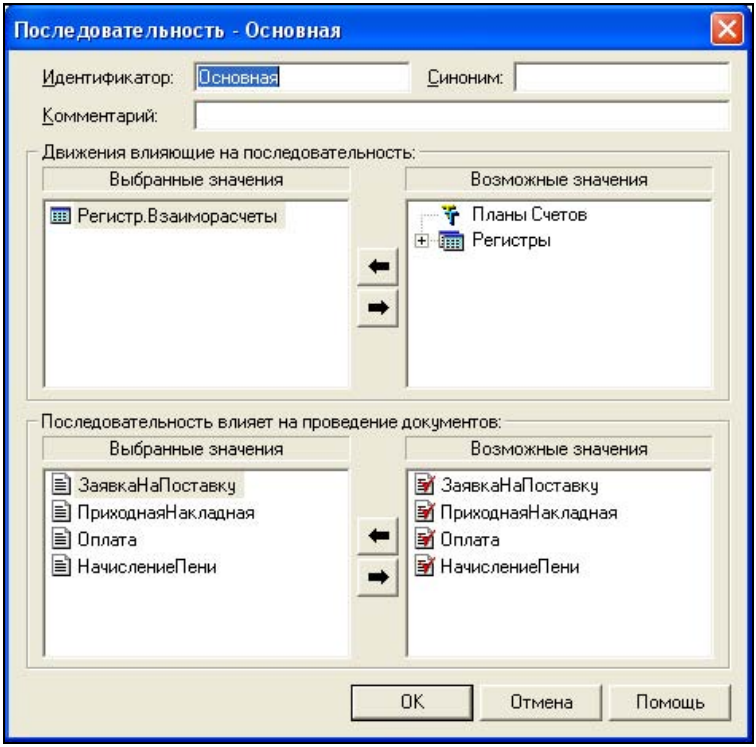


Рис. 10.11. Последовательность "Основная"

Задание 10.11

Разработайте последовательность "Заявки".

Пишем отчеты

Вся информация, необходимая для формирования отчетов, требуемых в задании, находится в регистре **Взаиморасчеты**. Поэтому алгоритм формирования

отчетов заключается в выборке итогов из регистра **Взаиморасчеты** по необходимым группировкам и выводе результатов запроса в таблицу отчета. Текст отчетов приведен в листингах 10.18 и 10.19.

Листинг 10.18. Отчет "График платежей"

```
Процедура ГрафикПлатежей()
        Перем Запрос, ТекстЗапроса, Таб;
        // Создание объекта типа "Запрос"
        Запрос = СоздатьОбъект("Запрос");
        ТекстЗапроса =
        "//{{ЗАПРОС(ГрафикПлатежей)
        |Период с ДатаОтчета по ДатаОтчета;
        |Срок = Регистр.Взаиморасчеты.Срок;
        |Поставщик = Регистр.Взаиморасчеты.Поставщик;
        |КредДокумент = Регистр.Взаиморасчеты.КредДокумент;
        |ВидДолга = Регистр.Взаиморасчеты.ВидДолга;
        |Сумма = Регистр.Взаиморасчеты.Сумма;
        |Функция СуммаПлатежа = КонОст(Сумма) Когда (Сумма>0);
        |Группировка Срок;
        |Группировка Поставщик;
        |Группировка КредДокумент;
        |Условие(ВидДолга = Перечисление.ВидыДолга.Оплата);
        //|Условие(Срок >= ДатаОтчета);
        |"//}}ЗАПРОС
        ;
        // Если ошибка в запросе, то выход из процедуры
        Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
                Возврат;
        КонецЕсли;
        Таб = СоздатьОбъект("Таблица");
        Таб.ИсходнаяТаблица("ГрафикПлатежей");
        Таб.ВывестиСекцию("Заголовок");
        Состояние("Заполнение выходной таблицы...");
        Таб.Опции(0, 0, Таб.ВысотаТаблицы(), 0);
        Пока Запрос.Группировка(1) = 1 Цикл
                Таб.ВывестиСекцию("Срок");
                Пока Запрос.Группировка(2) = 1 Цикл
                        Таб.ВывестиСекцию("Поставщик");
```

```
                    Пока Запрос.Группировка(3) = 1 Цикл
                        Таб.ВывестиСекцию("КредДокумент");
                    КонецЦикла;
                КонецЦикла;
        КонецЦикла;
        Таб.ВывестиСекцию("Итого");
        Таб.ТолькоПросмотр(1);
        Таб.Показать("ГрафикПлатежей", "");
КонецПроцедуры
```

Листинг 10.19. Отчет "Взаиморасчеты"

```
Процедура Взаиморасчеты()
        Запрос = СоздатьОбъект("Запрос");
        ТекстЗапроса= "";
        Если ВыбКонПериода>=ПолучитьДатуТА() Тогда
                ТекстЗапроса = ТекстЗапроса + "Период С ВыбНачПериода;";
        Иначе
                ТекстЗапроса = ТекстЗапроса + "Период С ВыбНачПериода По
ВыбКонПериода;";
        КонецЕсли;
        ТекстЗапроса = ТекстЗапроса+
        "//{{ЗАПРОС(Взаиморасчеты)
        |Поставщик = Регистр.Взаиморасчеты.Поставщик;
        |ВидДолга = Регистр.Взаиморасчеты.ВидДолга;
        |КредДокумент = Регистр.Взаиморасчеты.КредДокумент;
        |ТекущийДокумент = Регистр.Взаиморасчеты.ТекущийДокумент;
        |Сумма = Регистр.Взаиморасчеты.Сумма;
        |Функция СуммаНачОст = НачОст(Сумма);
        |Функция СуммаПриход = Приход(Сумма);
        |Функция СуммаРасход = Расход(Сумма);
        |Функция СуммаКонОст = КонОст(Сумма);
        |Группировка Поставщик;
        |Группировка ВидДолга;
        |Группировка КредДокумент;
        |Группировка ТекущийДокумент;
        |Условие(Поставщик в ВыбПоставщик);
        |"//}}ЗАПРОС
        ;
```

```
    // Если ошибка в запросе, то выход из процедуры
    Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
        Возврат;
    КонецЕсли;
    Таб = СоздатьОбъект("Таблица");
    Таб.ИсходнаяТаблица("Взаиморасчеты");
    Таб.ВывестиСекцию("Заголовок");
    Состояние("Заполнение выходной таблицы...");
    Таб.Опции(0, 0, Таб.ВысотаТаблицы(), 0);
    Пока Запрос.Группировка(1) = 1 Цикл
        Таб.ВывестиСекцию("Поставщик");
        Пока Запрос.Группировка(2) = 1 Цикл
            Таб.ВывестиСекцию("ВидДолга");
            Пока Запрос.Группировка(3) = 1 Цикл
                Таб.ВывестиСекцию("КредДокумент");
                Пока Запрос.Группировка(4) = 1 Цикл
                    Таб.ВывестиСекцию("ТекущийДокумент");
                КонецЦикла;
            КонецЦикла;
        КонецЦикла;
    КонецЦикла;
    Таб.ВывестиСекцию("Итого");
    Таб.ТолькоПросмотр(1);
    Таб.Показать("Взаиморасчеты", "");
КонецПроцедуры
```

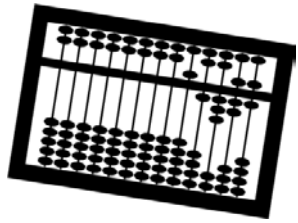
Задание 10.12

Проверьте правильность формирования движений модулей документов с помощью тестовых примеров, приведенных в табл. 10.3.

10.4. Контрольные вопросы

1. Каково назначение компоненты "Оперативный учет"?
2. Какие виды объектов метаданных доступны для конфигурирования в компоненте "Оперативный учет"?
3. Как в программе хранятся итоги? Каково назначение системы регистров?
4. Периодичность хранения остатков и влияние длительности периода на работу системы.

5. Какие данные и когда записываются в регистры?
6. В чем отличие регистров остатков от оборотных регистров?
7. Как можно построить выборку по движениям регистров? Как использовать значения реквизитов регистра для ограничения такой выборки?
8. Как организовать выборку по активным (ненулевым) остаткам регистра и обработать ее в цикле? В чем отличие обработки регистров остатков и оборотных регистров?
9. Что такое измерения, ресурсы и реквизиты регистра? Какие типы данных могут использоваться для их описания?
10. Какие вы знаете способы оптимизации выборки итогов и движений регистров? В каких случаях их целесообразно применять?
11. В чем отличие функций и процедур получения итогов по оборотным регистрам от функций и процедур получения остатков по регистрам остатков?
12. Что такое "временный расчет"? Когда его необходимо использовать? Какие методы встроенного языка для организации временного расчета вы знаете.
13. Перечислите особенности и назначение использования методов формирования движений регистров.
14. Как вы представляете себе структуру выборки в запросе по остаткам регистра? Чем она определяется? Что можно будет указывать в выражении "Запрос.ххххххх" после успешного выполнения такого запроса?
15. Когда наличие секции "Функция" в тексте запроса является обязательным для получения непустой выборки?



Глава 11

Работа со служебными типами данных и объектами компоненты "Расчет"

Компонента "Расчет" предназначена для решения задач по выполнению сложных периодических расчетов. Классическим примером таких задач является расчет заработной платы. Отметим и другие задачи: расчет суммы амортизации основных средств или нематериальных активов, расчет процентов по займам и кредитам, расчет квартплаты, расчет дивидендов по акциям.

Можно ли обойтись без объектов компоненты "Расчет" при расчете заработной платы? Можно, те же самые функции реализованы и в типовой конфигурации "Бухгалтерский учет", и в конфигурации для расчета заработной платы, разработанной калужской фирмой "Камин", в которых используются объекты только компоненты "Бухгалтерский учет". Однако применение специализированной компоненты позволяет использовать специальные механизмы, которые необходимы при реализации сложных периодических расчетов.

Что же особенное в компоненте "Расчет"? Прежде всего, есть механизм хранения записей расчета — **Журнал расчетов**. Записи хранятся по периодам, периодичность журнала одна из пяти: день, неделя, месяц, квартал, год. Каждая запись журнала расчетов имеет *период действия* и *период регистрации*. Эти периоды могут совпадать, если запись введена и действует в текущем периоде, и отличаться, если запись действует на прошлый или на будущий период.

Важной характеристикой записи журнала расчетов является **Вид расчета**. Виды расчета задаются в конфигураторе. У вида расчета есть модуль расчета, в котором описывается алгоритм проведения расчета. Вид расчета имеет *приоритет* — число от 0 до 999. Записи в журнале расчетов упорядочиваются по возрастанию приоритета видов расчета.

Одни виды расчета могут вытесняться другими. Это означает, что эти виды расчета не могут действовать одновременно: например, сотрудник не может одновременно получать оплату по окладу и оплату по больничному листу. Система автоматически выполняет *механизм вытеснения*. При вводе записи вытесняющего расчета записи вытесняемых расчетов уменьшаются по продолжительности или разбиваются на две записи (рис. 11.1).

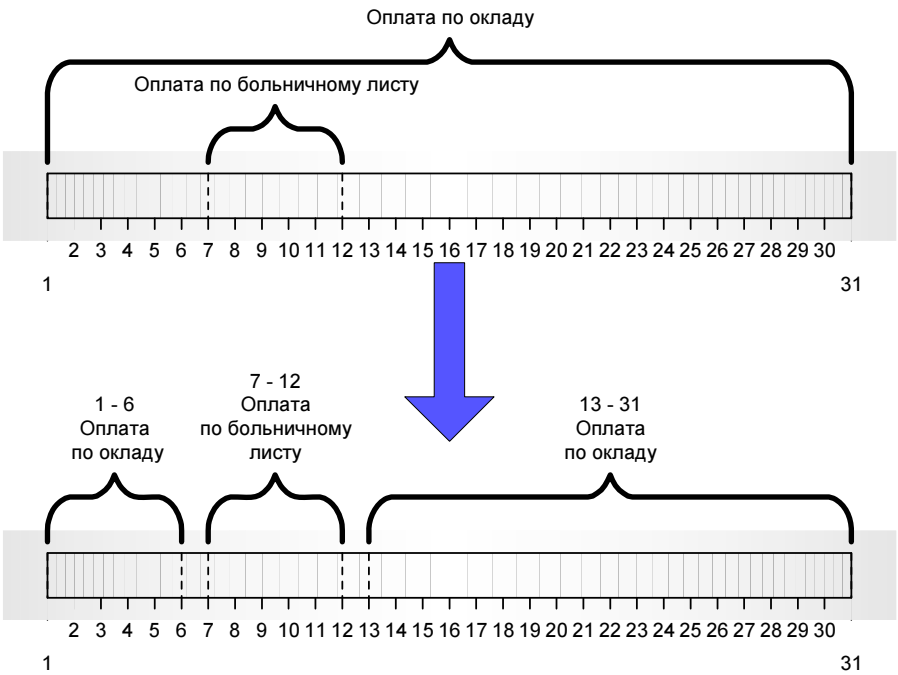


Рис. 11.1. Механизм вытесняющего расчета

Другим важным механизмом компоненты "Расчет" является *механизм перерасчетов*. Если два вида расчета связаны, как *ведущий* и *зависимый* (с помощью правила *перерасчета*), то при изменении результата записи ведущего расчета запись зависимого расчета делается нерассчитанной, а ее результат обнуляется. Также есть особенности при выполнении перерасчетов прошлых периодов. Для расчетов нет механизма, аналогичного бухгалтерским итогам, поэтому основным способом извлечения результатов расчетов является универсальный механизм запросов.

11.1. Объекты компоненты "Расчет"

Объектами компоненты "Расчет" являются календари, виды расчетов, группы и журналы расчетов.

11.1.1. Календарь

Календарь — это средство для хранения информации о рабочих днях. Для создания календаря необходимо открыть конфигурацию, встать на раздел метаданных **Календари**, нажать правую кнопку мыши и выбрать пункт **Новый Календарь**. В форме, представленной на рис. 11.2, необходимо заполнить название календаря, а на вкладке **Календарь** нужно задать информацию для автозаполнения календаря: стартовую дату и количество рабочих часов за один период. Так, для календаря-пятидневки надо задать стартовую дату — любой понедельник, пять дней по восемь часов и два по ноль часов, как показано на рис. 11.3.

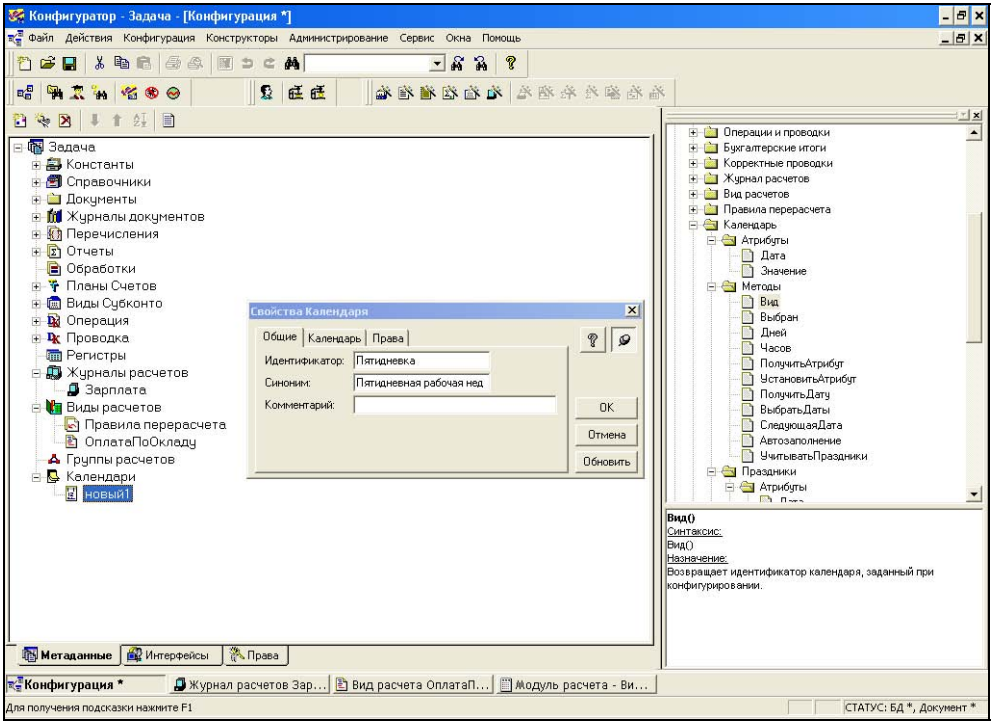


Рис. 11.2. Создание календаря

Чтобы выполнить автозаполнение календаря, нужно запустить 1С:Предприятие 7.7 в пользовательском режиме, в меню **Операции** выбрать пункт **Календари**, найти необходимый календарь, нажать кнопку **Автозаполнение** и указать период, за который необходимо заполнить календарь (рис. 11.4). Для ручного заполнения календаря нужно выбрать с помощью мыши день и ввести количество рабочих часов в этом дне. При этом количество дней определяется как количество дней с ненулевой продолжительностью рабочего дня, а количество часов — как сумма рабочих часов за заданный интервал времени.

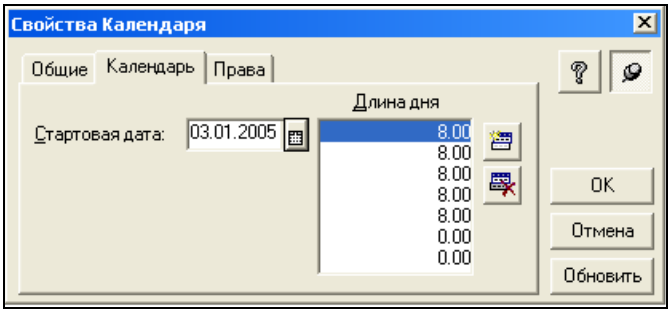


Рис. 11.3. Настройка календаря для автозаполнения

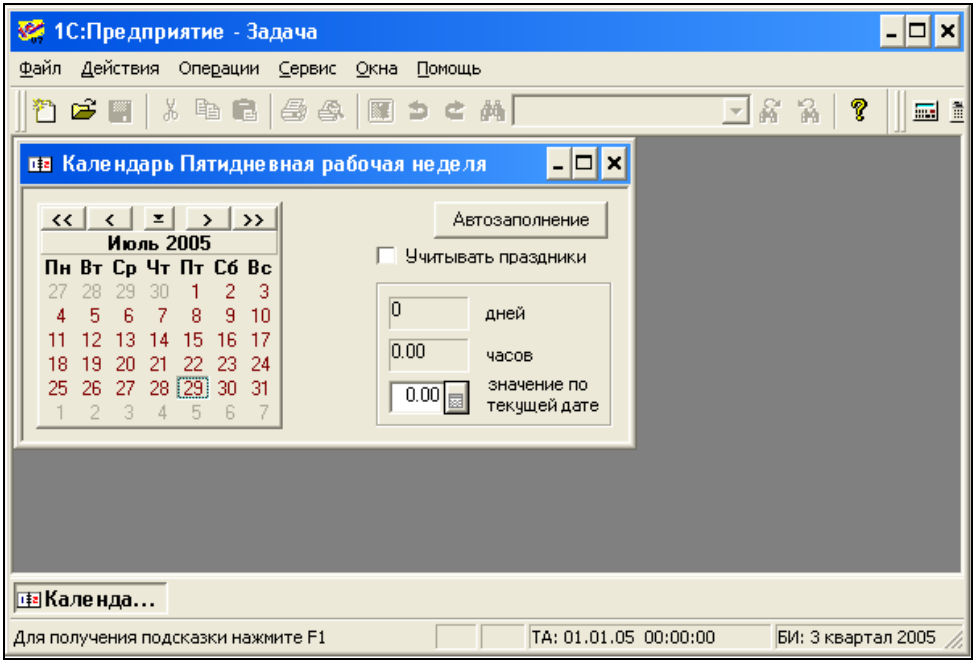


Рис. 11.4. Автозаполнение календаря

Если рабочие и выходные дни имеют какую-либо периодичность (например, пятидневная рабочая неделя — пять рабочих дней по восемь часов, два выходных), то праздничные дни не подчиняются какому-либо правилу, и их необходимо ввести в пользовательском режиме 1С:Предприятия 7.7. В меню **Операции** нужно выбрать пункт **Праздники** и задать "особые" дни, когда продолжительность рабочего дня либо равна нулю, либо сокращена на один час (рис. 11.5). Для того чтобы праздничные дни учитывались при автозаполнении, нужно установить флажок **Учитывать праздники**.

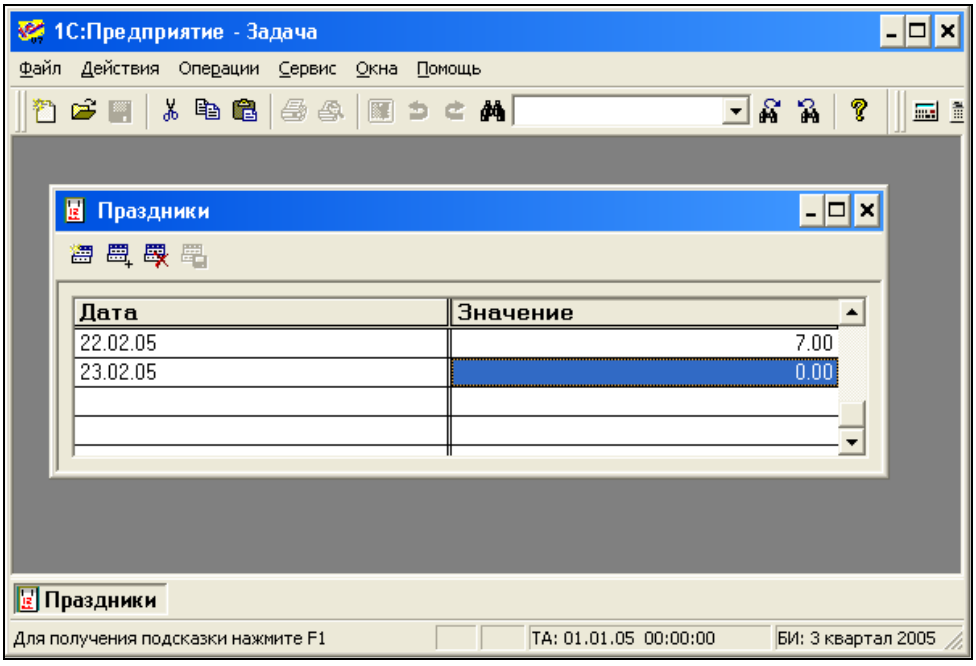


Рис. 11.5. Ввод праздничных дней

Календарь может быть типом константы, реквизита справочника, документа. Для доступа к календарю по идентификатору используется системная константа `Календари`. Объект типа "Календарь" обладает двумя системными атрибутами: `Дата` и `Значение`. В листинге 11.1 приведен пример установки значений календаря-пятидневки за период с `НачДата` по `КонДата` (реквизиты `НачДата` и `КонДата` задаются в форме диалога). Функция `Автозаполнение()` производит заполнение календаря датами (это единственный способ ввести даты в календарь программно!), а затем мы задаем конкретные значения рабочего времени в эти даты.

Листинг 11.1. Заполнение календаря программным способом

```
Календарь=Календари.Пятидневка;
Календарь.Автозаполнение(НачДата,КонДата);
Календарь.ВыбратьДаты(НачДата,КонДата);
Пока Календарь.СледующаяДата()=1 Цикл
        ТекДата=Календарь.Дата;
        ДеньНедели=НомерДняНедели(ТекДата);
        Если (ДеньНедели=6) или (ДеньНедели=7) Тогда
                Календарь.УстановитьАтрибут("Значение",0);
```

```
        Иначе
                Календарь.УстановитьАтрибут("Значение",8);
        КонецЕсли;
КонецЦикла;
```

Чтобы получить продолжительность рабочего времени за некоторый период, используются функции календаря `Дней(НачДата,КонДата)` и `Часов(НачДата,КонДата)`. Пример приведен в листинге 11.2.

Листинг 11.2. Пример работы с календарями

```
Сообщить("Количество рабочих дней по календарю пятидневной рабочей недели
в январе 2005 года составляет " +
Календари.Пятидневка.Дней('01.01.2005','31.01.2005'));
Сообщить("Количество рабочих часов по календарю пятидневной рабочей
недели в январе 2005 года составляет " +
Календари.Пятидневка.Часов('01.01.2005','31.01.2005'));
```

Для работы с праздниками существует специальный агрегатный тип `Праздники`. Чтобы программно ввести новые праздничные дни, используется функция `Новый()`. Пример ввода новых праздничных дней приведен в листинге 11.3.

Листинг 11.3. Пример работы с объектом "Праздники"

```
Пр=СоздатьОбъект("Праздники");
Пр.Новый('23.02.2005',0);
Пр.Новый('07.03.2005',7);
Пр.Новый('08.03.2005',0);
Пр.ВыбратьДаты('01.01.2005','31.12.2005');
Пока Пр.СледующаяДата()=1 Цикл
        Если Пр.Значение=0 Тогда
                Сообщить("Праздник "+Пр.Дата);
        Иначе
                Сообщить("Предпраздничный день "+Пр.Дата);
        КонецЕсли;
КонецЦикла;
```

Задание 11.1

1. Создайте новую (пустую конфигурацию).
2. Создайте календарь пятидневной рабочей недели.
3. Выполните автозаполнение календаря с 01.01.2005.
4. Напишите процедуру ввода праздничных дней за 2006 год.

11.1.2. Виды и группы расчетов

Чтобы добавить новый вид расчета, нужно открыть конфигурацию, выбрать с помощью мыши раздел метаданных **Виды Расчетов**, нажать ее правую кнопку и найти пункт **Новый Вид расчета**. В форме настройки вида расчета (рис. 11.6) задаем название вида расчета, принадлежность группам расчетов, приоритет. По кнопке **Настройка вытеснения** задаем вытесняющие и вытесняемые виды расчета.

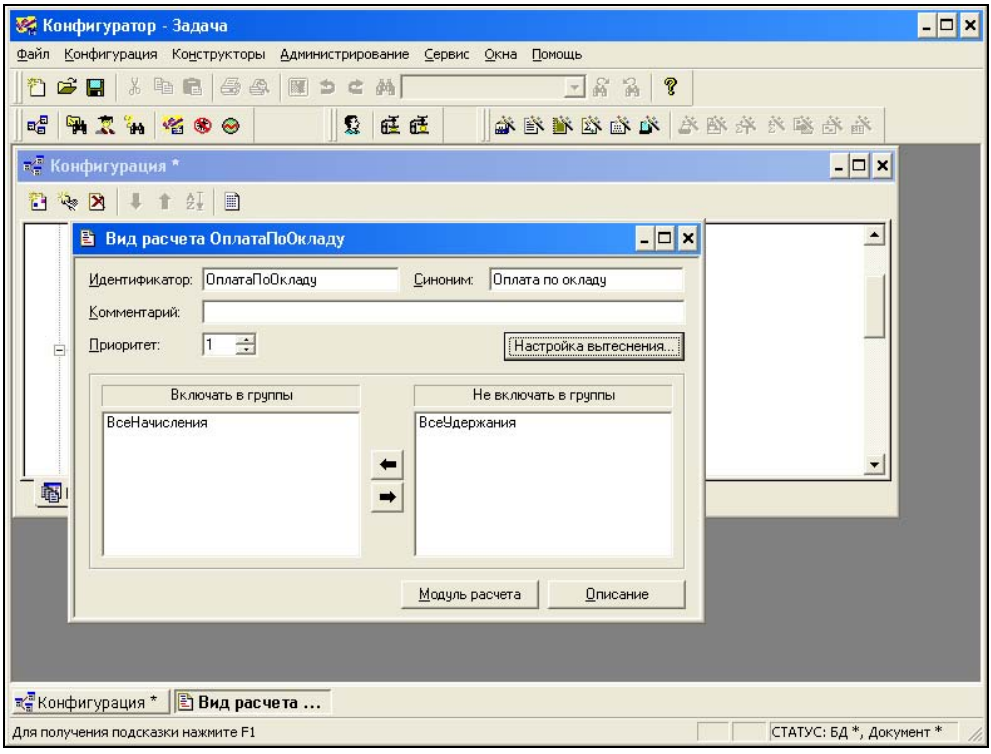


Рис. 11.6. Настройка вида расчета "Оплата по окладу"

Чтобы добавить группу расчетов, необходимо в дереве метаданных навести курсор мыши на раздел **Группы Расчетов**, нажать правую кнопку и выбрать пункт **Новая группа расчетов**. В форме настройки группы расчетов нужно перебросить в левый список из правого виды расчета, входящие в данную группу.

Один вид расчета может принадлежать нескольким группам расчетов.

С помощью групп расчетов можно определять *базу* расчета. Например, базой для расчета районного коэффициента являются все начисления за

исключением различных выплат социального характера (оплата больничного листа, стипендии и др.).

Для работы с видами и группами расчетов на встроенном языке используются системные константы `ВидРасчета` и `ГруппаРасчетов`. Например, чтобы получить доступ к виду расчета `ОплатаПоОкладу`, мы пишем `ВидРасчета.ОплатаПоОкладу`.

У вида расчета есть следующие реквизиты: `Код`, `Наименование`, `Очередность`.

- `Код` — идентификатор вида расчета;
- `Наименование` — комментарий к виду расчета;
- `Очередность` — приоритет вида расчета, задаваемый в конфигураторе.

Для определения принадлежности вида расчета определенной группе используется функция `ВходитВГруппу()`. А для определения вытеснения одного расчета другим используются функции `ВытесняетВидРасчета()` и `ВытесняетсяВидомРасчета()`. Примеры использования этих функций приведены в листингах 11.4—11.6. Для перебора всех видов расчета используется доступ к метаданным.

Листинг 11.4. Перебор всех видов расчета и вывод в окно сообщений, принадлежащих группе `ВсеНачисления`

```
Для Ном=1 По Метаданные.ВидРасчета() Цикл
        НазваниеВР= Метаданные.ВидРасчета(Ном);
        ВР= ВидРасчета.ПолучитьАтрибут(НазваниеВР);
        Если ВР.ВходитВГруппу(ГруппаРасчетов.ВсеНачисления)=1 Тогда
                Сообщить("Расчет "+ВР.Код+" ("+
                        ВР.Наименование+") входит в группу "+
                        ГруппаРасчетов.ВсеНачисления.Наименование+
                        );
        КонецЕсли;
КонецЦикла;
```

Листинг 11.5. Перебор всех видов расчета и вывод в окно сообщений, вытесняемых видом расчета `ВидРасч`

```
Процедура ВывестиСписокВытесняемых(ВидРасч)
Для Ном=1 По Метаданные.ВидРасчета() Цикл
        НазваниеВР= Метаданные.ВидРасчета(Ном);
        ВР= ВидРасчета.ПолучитьАтрибут(НазваниеВР);
        Если ВР.ВытесняетсяВидомРасчета(ВидРасч)=1 Тогда
```

```
                Сообщить("Вид расчета "+ВР.Код+
                " вытесняется видом расчета "+
                ВидРасч.Код);
        КонецЕсли;
КонецЦикла;
КонецПроцедуры
```

Листинг 11.6. Перебор видов расчета группы `ВсеНачисления` и вывод вытесняемых видов расчета

```
Группа=ГруппаРасчетов.ВсеНачисления;
Для Ном=1 По Группа.Количество() Цикл
        ВР=Группа.ПолучитьРасчет(Ном);
        Сообщить(ВР.Код);
        ВывестиСписокВытесняемых(ВР);
КонецЦикла;
```

Задание 11.2

1. Создайте виды расчета:
   - `ОплатаПоОкладу`, вытесняет `ОплатуПоОкладу`, приоритет 10;
   - `Прогул`, вытесняет `ОплатуПоОкладу`, приоритет 20;
   - `Премия`, приоритет 30;
   - `НДФЛ` (Налог на доходы физических лиц), приоритет 40;
   - `Выплата`, приоритет 50.
2. Создайте группы расчетов:
   - `ВсеНачисления`, содержит виды расчета `ОплатаПоОкладу`, `Прогул`, `Премия`;
   - `ВсеУдержания`, содержит вид расчета `НДФЛ`;
   - `ВсеВыплаты`, содержит вид расчета `Выплата`;
   - `БазаПремии`, содержит вид расчета `ОплатаПоОкладу`;
   - `БазаНДФЛ`, содержит виды расчета `ОплатаПоОкладу` и `Премия`.
3. Напишите внешний отчет, выводящий список всех видов расчета в окно сообщений.

11.1.3. Журнал расчетов

Журнал расчетов — это средство для хранения записей расчета.

Журналов расчетов может быть несколько, например, журнал расчета зарплаты и журнал расчета налогов. Чтобы добавить журнал расчетов, нужно открыть конфигурацию, встать на раздел метаданных **Журналы расчетов**, на-

жать правую кнопку мыши и выбрать пункт **Новый Журнал расчетов**. В форме настройки журнала расчетов (рис. 11.7), необходимо определить справочник объектов расчета (для расчета заработной платы это сотрудники, для расчета амортизации — основные средства и т. д.), периодичность журнала расчетов (выбор из фиксированных значений: день, неделя, месяц, квартал и год), длину и точность результата расчета, список дополнительных реквизитов расчета.

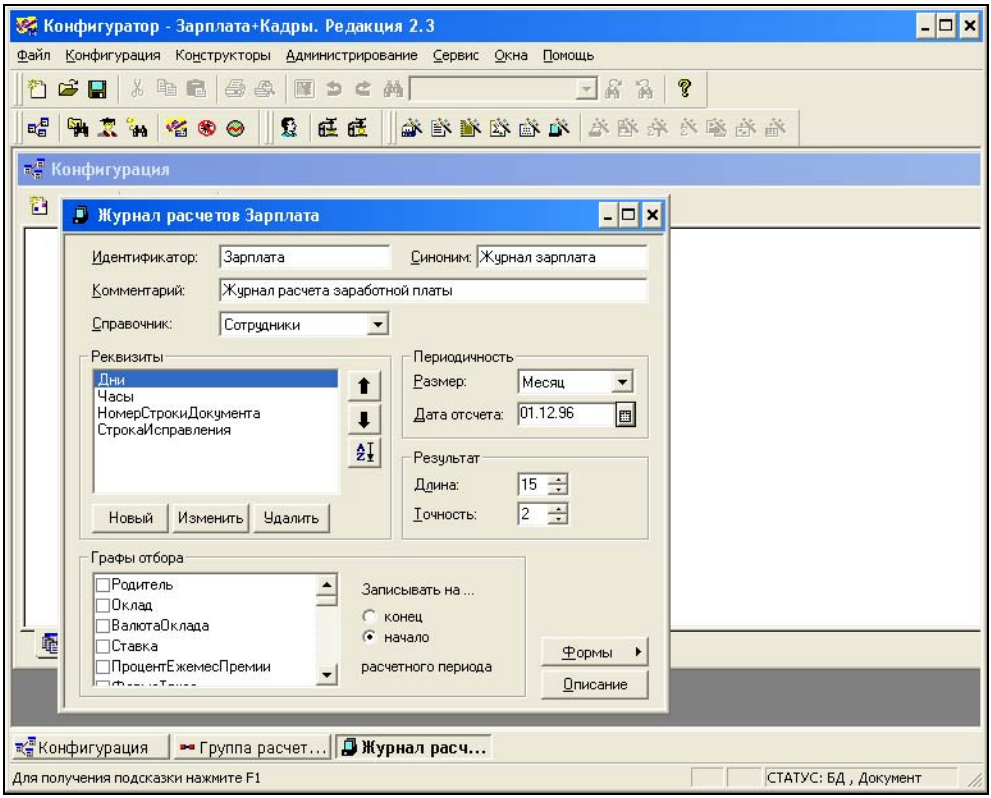


Рис. 11.7. Форма настройки журнала расчетов

Ввод записей в журнал расчетов можно выполнять только программным образом, но некоторые действия доступны при просмотре журнала расчетов. Чтобы открыть форму журнала расчетов в пользовательском режиме "1С:Предприятие 7.7", нужно выбрать меню **Операции**, пункт **Журналы расчетов** и необходимый журнал расчетов (рис. 11.8).

Через пункт меню **Действия** можно выполнить смену текущего периода, задать глубину просмотра журнала расчетов, выполнить запуск на проведение расчета (текущей записи журнала, всех записей текущего объекта, всех записей текущего документа), сделать ручное исправление результата расчета.

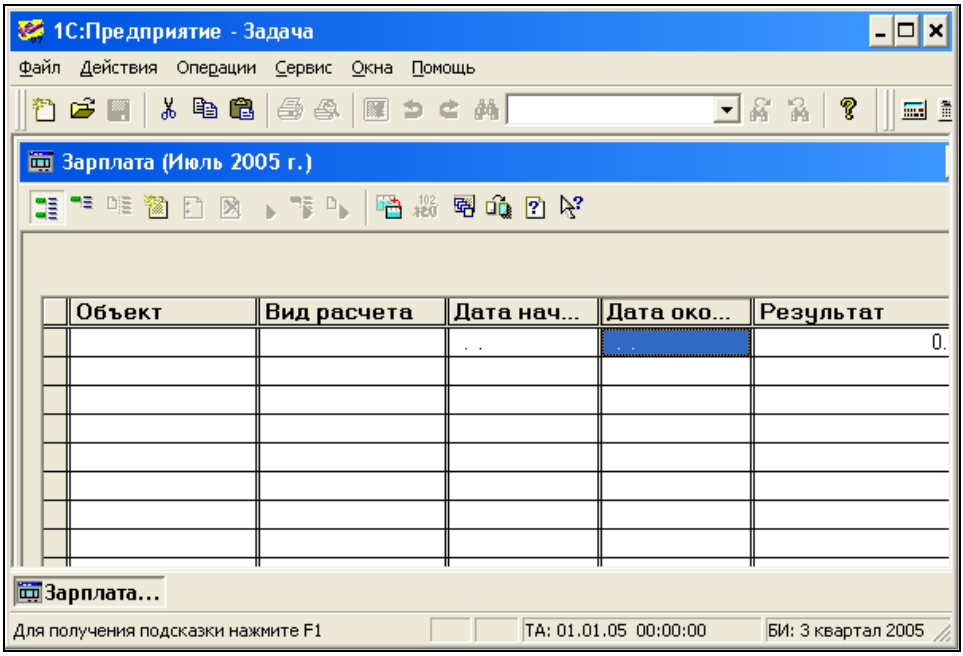


Рис. 11.8. Форма журнала расчетов

Задание 11.3

1. Создайте справочник "Сотрудники".
2. Создайте журнал расчетов Зарплата. В качестве объектов укажите справочник Сотрудники, периодичность — месяц, начало с 01.12.2004 г. Дополнительные реквизиты: Дни (тип Число), Часы (тип Число). Создайте форму журнала расчетов.

Реквизиты записи журнала расчетов

У записи журнала расчетов есть следующие системные реквизиты:

- РодительскийДокумент — ссылка на документ, который ввел запись в журнал расчетов;
- Документ — ссылка на документ — основание для ввода расчета;
- Объект — ссылка на элемент справочника — объект расчета;
- ВидРасч — вид расчета;
- ДатаНачала и ДатаОкончания — период действия расчета (не может превышать размер периодичности журнала расчетов);

- ПериодДействия — период, в котором действует запись (агрегатный тип — период журнала расчетов);
- ПериодРегистрации — период, в котором была зарегистрирована запись (агрегатный тип — период журнала расчетов);
- Результат — число, результат расчета записи.

Запись журнала расчетов имеет следующие признаки (число, принимающее значение 0 или 1) в зависимости от состояния расчета:

- Рассчитана — признак того, запись рассчитана (при отображении в журнале расчетов иконка содержит символ скрепки);
- Исправлена — признак того, что запись исправлена пользователем вручную (при отображении в журнале расчетов иконка имеет изображение руки);
- Фиксирована — признак того, что запись фиксирована (при отображении в журнале расчетов иконка имеет синий цвет). Фиксирование расчетов происходит при смене расчетного периода;
- Перерасчет — признак того, что запись является перерасчетом другой записи (ссылка на перерасчитываемую запись хранится в реквизите ПервичнаяЗапись);
- Сторно — признак того, запись является сторнирующей (обычно, первичная запись с минусом).

Реквизиты типа "Период журнала расчетов" имеют атрибуты ДатаНачала, ДатаОкончания, ОписательПериода и функцию ДобавитьПериод().

Для работы с журналом расчетов на встроенном языке используется агрегатный тип данных "ЖурналРасчетов". Чтобы создать переменную этого типа, нужно использовать функцию СоздатьОбъект():

```
ЖР=СоздатьОбъект("ЖурналРасчетов.Зарплата");
```

Текущий период журнала расчетов

У журнала расчетов есть понятие текущего периода. Все записи, которые вводятся в текущий момент (независимо от даты документа!), имеют период регистрации, совпадающий с текущим периодом.

В форме открытого журнала расчетов текущий период отображается в заголовке окна (см. рис. 11.8). Для смены текущего периода надо выбрать в меню **Действия** пункт **Сменить период расчета**, и в открывшейся форме (рис. 11.9) открыть следующий период или по кнопке **Дополн.** указать текущим произвольный период (как прошлый, так и будущий).

Текущий период можно получить и установить программно, как показано в листинге 11.7.

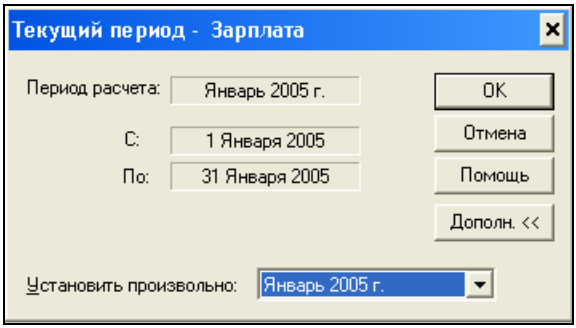


Рис. 11.9. Смена периода расчета зарплаты

Листинг 11.7. Работа с периодом журнала

```
ЖР=СоздатьОбъект("ЖурналРасчетов.Зарплата");
Сообщить("Текущий период: " + ЖР.НачалоТекущегоПериода()+ " - " +
ЖР.КонецТекущегоПериода());
НовыйПериод = ЖР.ПериодПоДате('01.01.2005');
ЖР.УстановитьТекущийПериод(НовыйПериод,0); // При смене периода не
                                            // выполняем системные действия
```

Внимание!

При смене в обратную сторону (в прошлый период) периода расчета заработной платы в режиме "1С:Предприятие" система автоматически делает все записи прошлого периода нерассчитанными и обнуляет результаты. Если нужно вернуться в прошлый период без потери результатов расчета, необходимо использовать второй параметр функции `УстановитьТекущийПериод()`, равный нулю, чтобы не выполнялись системные действия.

Задание 11.4

1. Установите вручную текущий период журнала расчетов `Зарплата` — январь 2005 г.
2. Напишите обработку `СменаПериода`, выполняющую смену текущего периода программно на заданную в форме диалога дату без выполнения системных действий.

Ввод расчета

Ввод записей в журнал расчета может производиться при проведении документа, у которого стоит признак принадлежности к компоненте "Расчет", в предопределенной процедуре `ОбработкаПроведения()`, а также при проведении расчета в процедуре `ПроведениеРасчета()` в модуле вида расчета.

Для ввода записи расчета используется функция ВвестиРасчет(<Объект>, <ВидРасчета>, <ДатаНачала>, <ДатаОкончания>, <Результат>).

Результат этой функции зависит от того, какие записи уже действуют в момент с <ДатаНачала> по <ДатаОкончания>, а также от того, в одном или в разных периодах находится <ДатаНачала> и <ДатаОкончания>.

Например, если ввести расчет, придставленный в листинге 11.8, то в журнал расчетов будет введено три записи (рис. 11.10): за январь, за февраль и за март.

Листинг 11.8. Ввод записи в журнал расчетов

```
ЖР=СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.ВвестиРасчет(Сотрудник,ВидРасчета.ОплатаПоОкладу,'01.01.2005',
'31.03.2005');
```

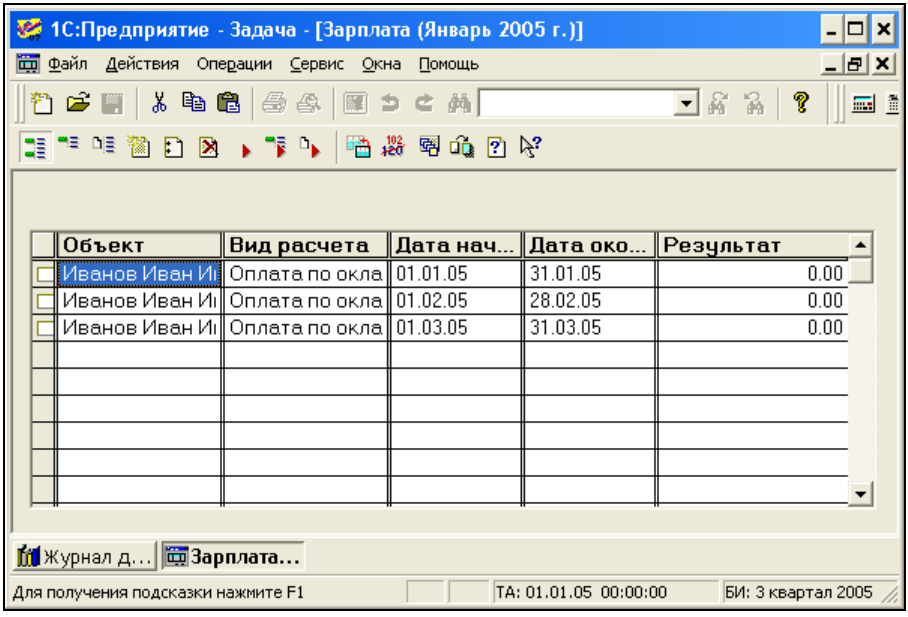


Рис. 11.10. Результат ввода расчета листинга 11.8

Если теперь ввести расчет "Прогул", вытесняющий расчет "ОплатаПоОкладу" (листинг 11.9), то в журнале расчетов появится пять записей (рис. 11.11). Первая запись была разбита на две: до прогула и после.

Листинг 11.9. Ввод записи журнала расчетов с вытеснением

```
ЖР=СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.ВвестиРасчет(Сотрудник,ВидРасчета.Прогул,'10.01.2005', 14.01.2005');
```

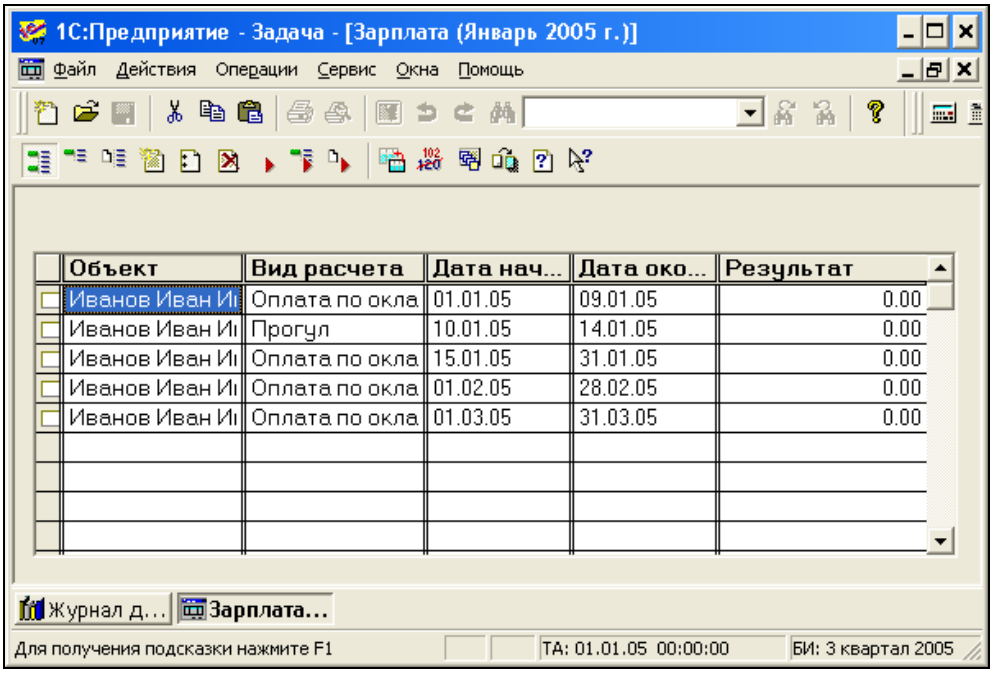


Рис. 11.11. Результат ввода расчета листинга 11.9

Особо следует отметить ситуацию, когда вытесняемые записи находятся не в текущем периоде. В этом случае записи не удаляются и не изменяются, а вводятся сторнирующие записи в текущем периоде.

Параметр `<Результат>` вводится в том случае, когда запись рассчитывать не нужно, и ее результат не зависит от продолжительности действия расчета или результатов других расчетов, то есть не требуется применение механизмов перерасчета и вытеснения.

Установку остальных реквизитов, которые не указываются в параметрах команды `ВвестиРасчет()`, можно выполнять предварительно командой `УстановитьРеквизит()`. По умолчанию реквизиты `Документ` и `РодительскийДокумент` инициализируются значением `ТекущийДокумент()`. Реквизит `ПериодДействия` автоматически определяется по параметрам `ДатаНачала` и `ДатаОкончания`. Реквизит `ПериодРегистрации` устанавливается равным текущему периоду журнала расчетов.

Если основанием расчета является какой-то другой документ, то вводить расчет нужно командой `ВвестиРасчетНаОсновании(<Основание>, <Объект>, <ВидРасчета>, <ДатаНачала>, <ДатаОкончания>, <Результат>)`, где `<Основание>` — это ссылка на документ-основание, который будет запомнен в реквизите `Документ` записи журнала расчетов.

Есть еще один способ добавить запись в журнал расчетов: с помощью функций `Новая()` и `Записать()`. В этом случае не выполняется механизм вытеснения и перерасчетов. Например, ввод расчета по выплате заработной платы документом `ВыплатаЗаработнойПлаты`, приведенный в листинге 11.10.

Листинг 11.10. Выплата заработной платы

```
ЖР.Новая();
ЖР.УстановитьРеквизит("Объект", Сотрудник);
ЖР.УстановитьРеквизит("ВидРасч", ВидРасчета.Выплата);
ЖР.УстановитьРеквизит("Документ", ТекущийДокумент());
ЖР.УстановитьРеквизит("РодительскийДокумент", ТекущийДокумент());
ЖР.УстановитьРеквизит("ДатаНачала", НачМесяца(Период));
ЖР.УстановитьРеквизит("ДатаОкончания", КонМесяца(Период));
ЖР.УстановитьРеквизит("Результат", Сумма);
ЖР.УстановитьРеквизит("Фиксирована", 1);
ЖР.УстановитьРеквизит("Рассчитана", 1);
ЖР.Записать();
```

Задание 11.5

1. Создайте документ `НачислениеЗарплаты`, имеющий реквизиты: `Сотрудник` (тип `Справочник.Сотрудники`), `ВидРасч` (тип `ВидРасчета`), `ДатаНачала` (тип `Дата`), `ДатаОкончания` (тип `Дата`), `Величина` (тип `Число`). Установите принадлежность к компоненте "Расчет".
2. Напишите процедуру `ОбработкаПроведения()`, выполняющую ввод расчета по реквизитам документа.
3. Введите документы `НачислениеЗарплаты` аналогично листингам 11.8 и 11.9 и посмотрите полученные после проведения записи в журнале расчета.
4. Создайте документ `ВыплатаЗаработнойПлаты`, имеющий реквизиты `Сотрудник` (тип `Справочник.Сотрудники`), `Период` (тип `Дата`), `Сумма` (тип `Число`).

Проведение расчета

Запустить проведение расчета текущей записи можно вручную из формы журнала расчетов и программно — с помощью функции `ВыполнитьРасчет()`. При этом запускается предопределенная процедура `ПровестиРасчет()`, расположенная в модуле вида расчета.

При проведении расчета на основании информации из объекта расчета, документа (родительского документа или документа основания), даты начала,

даты окончания расчета нужно определить результат расчета и записать его в реквизит `Результат`. В листинге 11.11 приведен пример расчета оплаты по окладу по дням. Для этого мы вычисляем норму по календарю пятидневной рабочей недели, определяем количество отработанных рабочих дней по календарю пятидневной рабочей недели, берем сумму оклада из документа-основания и вычисляем результат. Одновременно с расчетом результата мы рассчитываем количество отработанных дней, часов и записываем их в дополнительные реквизиты.

Листинг 11.11. Расчет оплаты по окладу по дням

```
Процедура ПровестиРасчет()
        Норма=Календари.Пятидневка.Дней(НачМесяца(ДатаНачала),
        КонМесяца(ДатаОкончания));
        Дни=Календари.Пятидневка.Дней(ДатаНачала,ДатаОкончания);
        Часы=Календари.Пятидневка.Часов(ДатаНачала,ДатаОкончания);
        Если Норма<>0 Тогда
        // Величину оклада берем из документа
        Результат=Дни/Норма*Документ.Величина;
        КонецЕсли;
КонецПроцедуры
```

Задание 11.6

1. Напишите процедуры расчета видов `ОплатаПоОкладу`, `Прогул`.
2. Рассчитайте записи этих видов расчета в режиме "1С:Предприятие", проверьте правильность расчета.

Работа с записями журнала расчетов на встроенном языке

Чтобы извлечь информацию из журнала расчетов, можно перебрать записи журнала с помощью функций `ВыбратьЗаписи(НачДата,КонДата)` и `ПолучитьЗапись()`. При этом в выборку попадают все записи, период действия которых имеет пересечение с заданным интервалом с `НачДата` по `КонДата`. В листинге 11.12 приведен пример получения суммы всех начислений по заданному сотруднику.

Листинг 11.12. Сумма всех начислений по сотруднику

```
ЖР=СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.ВыбратьЗаписи(НачДата,КонДата);
Сум=0;
```

```
Пока ЖР.ПолучитьЗапись()=1 Цикл
        Если ЖР.Объект<>ВыбСотрудник Тогда
                Продолжить;
        КонецЕсли;
        Если ЖР.ВидРасч.ВходитВГруппу(ГруппаРасчетов.ВсеНачисления)=1
                Тогда
                Сум=Сум+ЖР.Результат;
        КонецЕсли;
КонецЦикла;
Сообщить("Сумма всех начислений сотрудника "+
        ВыбСотрудник.Наименование+" за период с "+НачДата+
        " по "+КонДата+" равна "+Сум);
```

Использование запросов при работе с записями журнала расчетов

Наиболее эффективным способом извлечения результатов из журнала расчетов является использование механизма универсальных запросов *(см. главу 5)*. Основная особенность использования запросов состоит в том, что есть два способа задания периода запроса. Действительно, каждая запись имеет два периода — период действия и период регистрации. Поэтому одну и ту же запись можно отнести к разным периодам.

Если в тексте запроса написать `Период С Д1 По Д2`, то в запрос будут попадать записи журнала расчетов по периоду регистрации, имеющему пересечение с интервалом дат [`Д1, Д2`]. Если написать `С Д1 По Д2`, то в запрос будут включены записи по периоду действия (реквизиты `ДатаНачала` и `ДатаОкончания` записи журнала расчетов), имеющему пересечение с интервалом дат [`Д1, Д2`].

Например, для получения результата, аналогичного результату выполнения программы листинга 11.12, можно написать запрос, приведенный в листинге 11.13.

Листинг 11.13. Вывод в окно сообщений суммы всех начислений по сотрудникам

```
Группа = ГруппаРасчетов.ВсеНачисления;
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(ЗапросПоНачислениям)
|С НачДата По КонДата;
|Объект = ЖурналРасчетов.Зарплата.Объект;
```

```
|ВидРасч = ЖурналРасчетов.Зарплата.ВидРасч;
|Результат = ЖурналРасчетов.Зарплата.Результат;
|Функция Сум = Сумма(Результат);
|Группировка Объект;
|Условие(ВидРасч.ВходитВГруппу(Группа)=1);
|"//}}ЗАПРОС
;
Запрос.Выполнить(ТекстЗапроса);
Пока Запрос.Группировка(1)=1 Цикл
        Сообщить("Сумма начислений сотрудника "+
        Запрос.Объект.Наименование+
        " за период с "+НачДата+ " по "+КонДата+
        " равна "+Запрос.Сум);
КонецЦикла;
```

Задание 11.7

Напишите процедуру вычисления суммы начислений двумя способами: перебором записей журнала расчетов и с помощью запроса.

11.2. Основные механизмы расчетов

Рассмотрим более подробно основные механизмы компоненты "Расчет", поскольку в начале главы о них было сказано кратко.

11.2.1. База расчетов

Часто бывает, что результат одного расчета R является функцией от результатов других расчетов R_1, R_2, ..., R_n, например,

$$R = F(R_1 + R_2 + ... + R_n).$$

Базой называется, с одной стороны, группа расчетов $\boldsymbol{R_1}$, $\boldsymbol{R_2}$, ..., $\boldsymbol{Rn}$, с другой стороны, — сумма результатов расчетов

$$\boldsymbol{B} = R_1 + R_2 + ... + R_n$$

за некоторый период.

Расчет премии процентом

Месячная премия вычисляется как процент от оклада:

$$Премия = Процент \times Оклад / 100.$$

Расчет премии приведен в листинге 11.14. Для получения базы используется запрос с выборкой по периоду действия.

Листинг 11.14. Расчет премии

```
Процедура ПровестиРасчет()
БазаПремии=ГруппаРасчетов.БазаПремии;
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(РасчетБазы)
|С ДатаНачала по ДатаОкончания;
|Сотрудник = ЖурналРасчетов.Зарплата.Объект;
|ВидРасч = ЖурналРасчетов.Зарплата.ВидРасч;
|Результат = ЖурналРасчетов.Зарплата.Результат;
|Функция База = Сумма(Результат);
|Группировка Сотрудник;
|Условие(ВидРасч.ВходитВГруппу(БазаПремии)=1);
|Условие(Сотрудник=Объект);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
        Возврат;
КонецЕсли;
Пока Запрос.Группировка(1) = 1 Цикл
        // Процент премии берем из документа
        Результат=Запрос.База*Документ.Величина/100;
КонецЦикла;
КонецПроцедуры
```

Расчет НДФЛ

Рассмотрим еще один пример — *расчет суммы налога на доходы физических лиц (НДФЛ)*. С доходов работников удерживается налог на доходы, исчисляемый как процентная доля налоговой базы (доходов работника). Процентная доля (ставка налога) в течение налогового периода не меняется. Исчисление суммы налога производится нарастающим итогом с начала налогового периода по итогам каждого месяца с зачетом суммы налога, удержанной в предыдущие месяцы текущего налогового периода.

Алгоритм расчета выглядит следующим образом:

1. Рассчитываем сумму дохода с начала года.
2. Вычисляем сумму налога:

$$Налог = Процент \times Доход\ с\ начала\ года\ /100.$$

3. Вычисляем сумму налога, начисленного по предыдущий месяц.
4. Суммой налога текущего месяца является разница между суммой налога с начала года и суммой налога, начисленного по предыдущий месяц.

Процедура расчета НДФЛ приведена в листинге 11.15. При реализации алгоритма сумму дохода с начала года и сумму налога, начисленного по предыдущий месяц, можно получить в одном запросе.

Листинг 11.15. Расчет НДФЛ

```
Процедура ПровестиРасчет()
БазаНДФЛ=ГруппаРасчетов.БазаНДФЛ;
НДФЛ=ВидРасчета.НДФЛ;
НачалоГода=НачГода(ДатаНачала);
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(РасчетБазы)
|С НачалоГода по ДатаОкончания;
|Сотрудник = ЖурналРасчетов.Зарплата.Объект;
|ВидРасч = ЖурналРасчетов.Зарплата.ВидРасч;
|ДатаОконч = ЖурналРасчетов.Зарплата.ДатаОкончания;
|Результат = ЖурналРасчетов.Зарплата.Результат;
|Функция База = Сумма(Результат) когда
↳(ВидРасч.ВходитВГруппу(БазаНДФЛ)=1);
|Функция НДФЛ = Сумма(Результат) когда ((ВидРасч=НДФЛ) и
↳(ДатаОконч<ДатаОкончания));
|Группировка Сотрудник;
|Условие(Сотрудник=Объект);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
        Возврат;
КонецЕсли;
Пока Запрос.Группировка(1) = 1 Цикл
        Результат=Запрос.База*Константа.ПроцентНДФЛ/100-Запрос.НДФЛ;
КонецЦикла;
КонецПроцедуры
```

Задание 11.8

1. Создайте константу `ПроцентНДФЛ` и задайте ее значение в режиме "1С:Предприятие".
2. Создайте модули видов расчета `Премия` и `НДФЛ`. Протестируйте их работу как минимум на двух расчетных периодах.

11.2.2. Механизм вытеснения

Вытеснение одних расчетов другими настраивается в форме **Порядок вытеснения** (рис. 11.12), которая вызывается из формы настройки вида расчета.

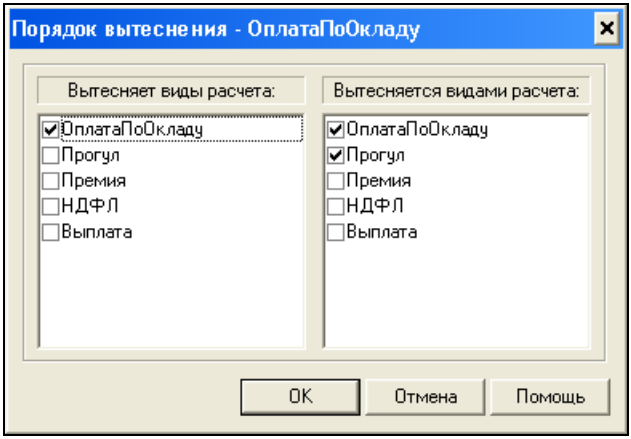


Рис. 11.12. Форма **Порядок вытеснения**

Вид расчета может вытеснять сам себя. Это действительно имеет смысл, если одновременно не могут действовать две записи одного и того же вида расчета. В этом случае установка вытеснения вида расчета самого себя предупреждает возможность ошибочного ввода повторной записи.

Операция вытеснения не является транзитивной. Если вид расчета R_1, вытесняет вид расчета R_2 , а вид расчета R_2 вытесняет вид расчета R_3, то из этого не следует, что вид расчета R_1 вытесняет вид расчета R_3. Даже наоборот, может быть ситуация, когда вид расчета R_3 вытесняет вид расчета R_1.

При вводе вытесняющего расчета с периодом действия с `ДатаНачала` по `ДатаОкончания` проводится анализ уже существующих записей журнала расчета:

- если в этом периоде уже существуют записи с вытесняемыми видами расчета, будет проводиться редактирование периода их действия (в том числе разбиение на две записи) или полное их удаление;

- если существуют записи, вытесняющие вводимый расчет, то корректируется период действия вводимой записи (в том числе разбиения на две записи). Это может привести, например, к тому, что новая запись вовсе не будет введена, если ее место во времени уже занято вытесняющими записями.

Из рассмотренного механизма следует, что при *взаимном вытеснении* двух или нескольких видов расчета большое значение имеет порядок ввода записей этих видов расчета.

Задание 11.9

Пусть есть три вида расчета:

- R_1 вытесняет вид расчета R_2,
- R_2 вытесняет вид расчета R_3,
- R_3 вытесняет вид расчета R_1.

Последовательно вводятся следующие записи:

- R_1 с 01.02.2005 по 28.02.2005;
- R_3 с 01.02.2005 по 10.02.2005;
- R_2 с 05.02.2005 по 15.02.2005.

Какие записи в результате будут в журнале расчетов?

Вытеснение записей прошлого периода

В том случае, если вытесняемые данные принадлежат не текущему периоду расчета, проводится не корректировка или полное удаление записей, а их сторнирование (признак `Сторно` равен единице, а результат равен результату вытесняемой записи с минусом). Сторнирующую запись надо рассчитывать, расчет выполняется по тому же алгоритму, только результат будет со знаком минус.

11.2.3. Механизм перерасчетов

Механизм перерасчетов возникает в том случае, если при изменении данных одних расчетов нужно пересчитать другие. Для создания перерасчета нужно выделить подраздел **Правила перерасчета** раздела **Виды расчета** метаданных, нажать правую кнопку мыши и выбрать пункт **Новое Правило перерасчета**. В левом списке (рис. 11.13) нужно отметить *ведущие* виды расчета, в правом списке — *зависимые* виды расчета.

Правило перерасчета может быть двух типов: перерасчет "по текущему периоду" или "по будущим периодам". В первом случае перерасчитываются заданные виды расчетов с тем же периодом действия, что и новая запись. Во втором — перерасчитываются записи одного или нескольких будущих

расчетных периодов. Количество будущих периодов также указывается в соответствующем поле на форме настройки перерасчета.

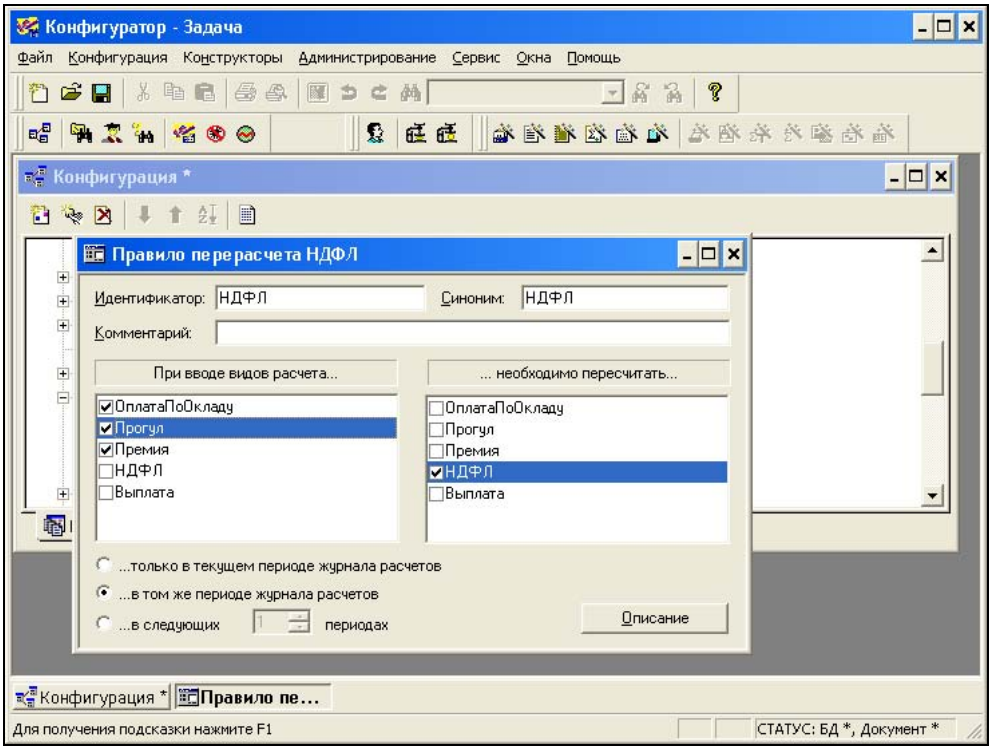


Рис. 11.13. Настройка правила перерасчета

Как работает механизм перерасчета

Механизм перерасчета работает следующим образом.

Итак, если в журнале расчетов в текущем периоде появится новая (в результате проведения документа), исчезнет (при отмене проведения) или будет исправлена существующая запись с одним из ведущих видов расчета, то будет снят признак "Рассчитана" со всех записей, соответствующих зависимым видам расчета, с тем же периодом действия, что и введенная, удаленная или исправленная запись.

Если при этом вводится запись с периодом действия не в текущем расчетном периоде, а в одном из прошлых (например, расчет оклада задним числом за прошлый месяц), то система автоматически введет записи-перерасчеты для всех зависимых видов расчета соответствующего прошлого периода.

Задание 11.10

1. Создайте правило перерасчета `НДФЛ` согласно рис. 11.13. Посмотрите результат действия механизма перерасчетов при перепроведении документа "`НачислениеЗарплаты`", при ручном изменении результата расчета оплаты по окладу или премии.
2. Создайте правило перерасчета `Премии`.

Расчет оплаты по больничному листу

Рассмотрим пример, наглядно показывающий применение правила перерасчета будущих периодов. При расчете оплаты больничного листа рассчитывается средний заработок за предыдущие двенадцать месяцев. Таким образом, при изменении расчета, входящего в базу оплаты по больничному листу, необходимо пересчитать оплату по больничному листу за последующие 12 периодов (рис. 11.14). Модуль расчета оплаты по больничному листу приведен в листинге 11.16.

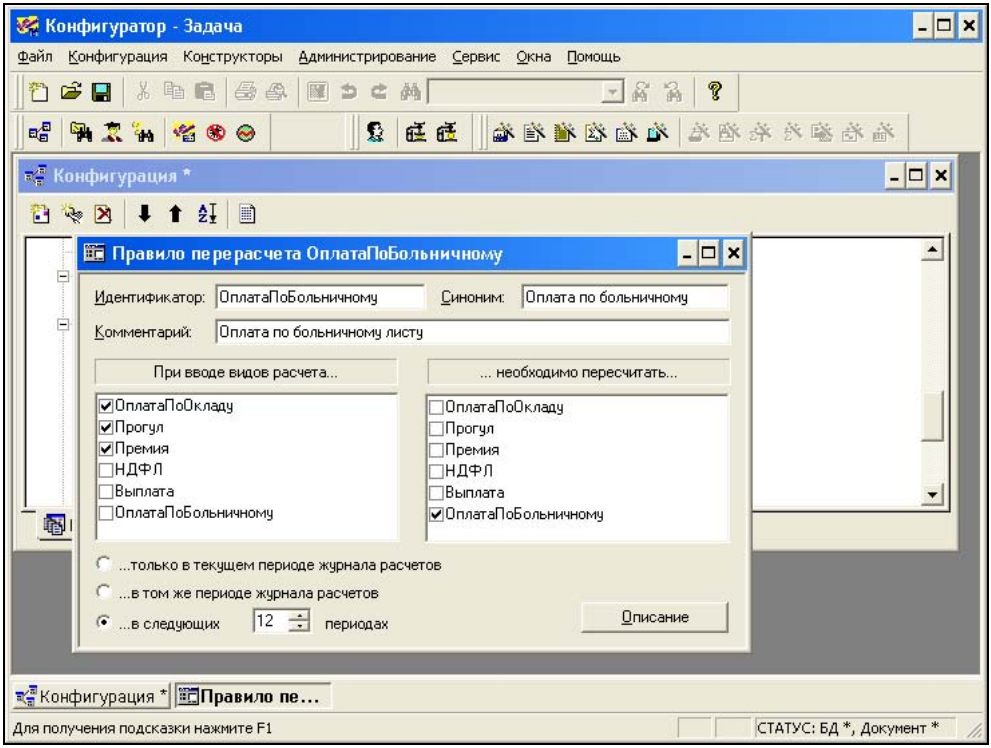


Рис. 11.14. Правило перерасчета `ОплатаПоБольничному`

Листинг 11.16. Расчет оплаты по больничному листу

```
Процедура ПровестиРасчет()
БазаБольничного=ГруппаРасчетов.БазаБольничного;
Начало=ДобавитьМесяц(НачМесяца(ДатаНачала),-12);
Конец=ДобавитьМесяц(КонМесяца(ДатаОкончания),-1);
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(РасчетБазы)
|С Начало по Конец;
|Сотрудник = ЖурналРасчетов.Зарплата.Объект;
|ВидРасч = ЖурналРасчетов.Зарплата.ВидРасч;
|ДатаОконч = ЖурналРасчетов.Зарплата.ДатаОкончания;
|Результат = ЖурналРасчетов.Зарплата.Результат;
|Дни = ЖурналРасчетов.Зарплата.Дни;
|Функция База = Сумма(Результат);
|Функция СуммаДней = Сумма(Дни);
|Группировка Сотрудник;
|Условие(Сотрудник=Объект);
|Условие(ВидРасч.ВходитВГруппу(БазаБольничного)=1);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
        Возврат;
КонецЕсли;
Пока Запрос.Группировка(1) = 1 Цикл
        Если Запрос.СуммаДней>0 Тогда
                СреднедневнойЗаработок=Запрос.База/Запрос.СуммаДней;
                // Определяем количество дней на больничном по календарю
                Дни=Календари.Пятидневка.Дней(ДатаНачала,ДатаОкончания);
                Результат=СреднедневнойЗаработок*Дни;
        КонецЕсли;
КонецЦикла;
КонецПроцедуры
```

Задание 11.11

1. Создайте вид расчета `ОплатаПоБольничному`.
2. Создайте группу расчетов `БазаБольничного`.

3. Создайте правило перерасчета `ОплатаПоБольничному`.
4. Протестируйте работу расчета оплаты по больничному листу на ряде примеров.

11.3. Пример решения экзаменационной задачи

При решении задачи аттестационного экзамена по компоненте "Расчет", проводимого фирмой "1С", необходимо выполнять предъявляемые требования, опубликованные на сайте **www.1c.ru**.

11.3.1. Требования к решению задачи

К решению задачи аттестационного экзамена предъявляются следующие требования:

1. Конфигурация для решения поставленной задачи создается "с нуля", время на решение дается до 4 часов.
2. Оперативная информация, необходимая для расчетов, не должна вноситься через справочники (например, через периодические реквизиты), а должна попадать в систему через документы.
3. Информация, влияющая на результаты расчета, должна из документов попадать в журнал расчетов.
4. При решении задачи обязательно использование специфических объектов метаданных компоненты "Расчет":
   - Журнал расчетов;
   - Вид расчетов;
   - Группа расчетов;
   - Календарь.
5. При расчете результатов записей журнала расчетов обязательно использование его атрибутов (хотя бы один пример использования):
   - Вид расчета;
   - Дата начала;
   - Дата окончания;
   - Результат;
   - Родительский документ.
6. Если информация, необходимая для расчета результата записи, присутствует в журнале расчетов, ее следует извлекать оттуда, а не из документов или справочников.
7. При написании модулей необходимо учитывать скорость, с которой будут выполняться написанные процедуры. Например: существуют 200 до-

кументов по 100 сотрудникам, в многострочной части этих документов содержится информация о виде выполненных каждым сотрудником работ. Поиск перечня работ, выполненных сотрудником, следует проводить с использованием запросов или графы отбора, а не циклами по всем документам и их строкам.

11.3.2. Задача "Расчет зарплаты работников ЧОП"

ЧОП — это частное охранное предприятие. В табл. 11.1 приведен примерный текст задания с комментариями.

Таблица 11.1. *Текст задания с комментариями*

№	Текст задания	Комментарий
I.	Создайте оригинальную конфигурацию, предусматривающую возможность расчетов по оплате труда работников организации, оказывающей услуги по охране объектов недвижимого имущества заказчиков, для следующих особенностей учета и оплаты труда	
I-1	Трудовая деятельность работников предприятия заключается в охране различных объектов недвижимого имущества заказчиков	1. Создать справочник "Сотрудники". 2. Создать справочник "ОбъектыОхраны"
I-2	Оплата труда по охране каждого объекта производится в соответствии с установленной часовой тарифной ставкой. Тарифные ставки по объектам различаются	3. Создать вид расчета ОплатаПоТарифу. 4. В справочник "ОбъектыОхраны" добавить реквизит Тариф (тип Число)
I-3	В течение месяца работник может работать на разных объектах	
I-4	Один раз в неделю расчетчик вводит в систему информацию о том, на каких объектах и сколько часов работал каждый работник	5. Создать документ "ИндивидуальныйНаряд" с реквизитами шапки — Сотрудник (тип Справочник.Сотрудники), реквизитами табличной части — ОбъектОхраны (тип Справочник.ОбъектыОхраны), — Часы (тип Число), — Сумма (тип Число) — произведение часов на тариф

Таблица 11.1 *(продолжение)*

№	Текст задания	Комментарий
I-5	Расчет заработной платы производится один раз в месяц	6. Создать журнал расчета Зарплата, объекты — справочник "Сотрудники", периодичность — месяц. Дополнительные реквизиты: ОбъектыОхраны (тип Справочник.ОбъектыОхраны), Часы (тип Число). 7. Создать документ "НачислениеЗарплаты"
I-6	За каждый день, не проработанный по графику рабочего времени по причине временной нетрудоспособности, работнику выплачивается пособие в размере 50% среднечасового заработка в месяце, предшествующем месяцу наступления нетрудоспособности, умноженном на 8	8. Создать Календари: График1, График2 (в задаче не указано по скольким графикам работают охранники, предположим, что их два). 9. У справочника "Сотрудники" создать реквизит График (тип Календарь), периодический, изменяется только документами. 10. Создать документ "СменаГрафика" с реквизитами шапки Сотрудник (тип Справочник.Сотрудники), График (тип Календари). В обработке проведения написать установку значения реквизита График справочника "Сотрудники", при отмене проведения снимать установку значения реквизита График. 11. Создать документ "БольничныйЛист" с реквизитами шапки: Сотрудник (тип Справочник.Сотрудники), ДатаНачала (тип Дата), ДатаОкончания (тип Дата). 12. Создать вид расчета ОплатаПоБольничному
I-7	При этом работнику, который замещает заболевшего коллегу, за каждый час замещения производится доплата в размере 50% среднечасового заработка за предыдущий месяц	13. Создать вид расчета ДоплатаЗаЗамещение. 14. Создать перечисление ТипыРаботы с двумя значениями: Обычный и Замещение. 15. В документ ИндивидуальныйНаряд добавить реквизит табличной части ТипРаботы (тип Перечисление.ТипыРаботы)

Таблица 11.1 *(окончание)*

№	Текст задания	Комментарий
II	Конфигурация должна содержать отчет, формируемый с использованием запросов и позволяющий за произвольное количество расчетных периодов получать информацию следующего характера:	16. Данные отчета будут получены запросом по периоду действия с тремя группировками – по сотрудникам, по объектам, по периодам действия.

	<Период 1>		**<Период 2>**	
	Проработано часов	**Сумма заработка**	**Проработано часов**	**Сумма заработка**
<Работник>				
в том числе:				
<Объект>				
Итого				

Анализируя текст задания, мы "на полях" записали, какие объекты метаданных необходимо создать.

Задание 11.12

Создайте новую (пустую) конфигурацию со следующими объектами метаданных.

1. Справочники:
   - "Сотрудники", реквизит График (тип Календарь), периодический, изменяется только документами;
   - "ОбъектыОхраны", реквизит Тариф (тип Число).
2. Календари: График1, График2.
3. Журнал расчета Зарплата, объекты — справочник "Сотрудники", периодичность — месяц, дополнительные реквизиты — Часы, ОбъектыОхраны.
4. Перечисление ТипыРаботы с двумя значениями: Обычный и Замещение.
5. Документы:
   - СменаГрафика с реквизитами шапки:
     - Сотрудник (тип Справочник.Сотрудники);
     - График (тип Календари).

В обработке проведения написать установку значения периодического реквизита `График` справочника `"Сотрудники"`, при отмене проведения снимать установку значения реквизита `График`.

- `ИндивидуальныйНаряд`, с реквизитами шапки:
  - `Сотрудник` (тип `Справочник.Сотрудники`);
- реквизитами табличной части:
  - `ОбъектОхраны` (тип `Справочник.ОбъектыОхраны`);
  - `Часы` (тип `Число`);
  - `Сумма` (тип `Число`) — произведение часов на тариф;
  - `ТипРаботы` (тип `Перечисление.ТипыРаботы`).
- `БольничныйЛист` с реквизитами шапки:
  - `Сотрудник` (тип `Справочник.Сотрудники`);
  - `ДатаНачала` (тип `Дата`);
  - `ДатаОкончания` (тип `Дата`).
- `НачислениеЗарплаты`.

6. Виды расчета:
   - `ОплатаПоТарифу`, приоритет 10;
   - `ДоплатаЗаЗамещение`, приоритет 20;
   - `ОплатаПоБольничному`, приоритет 30, вытесняет виды расчета `ОплатаПоТарифу`, `ДоплатаЗаЗамещение`.
7. Группы расчетов:
   - `БазаДляСреднего`, включает виды расчетов `ОплатаПоТарифу`, `ДоплатаЗаЗамещение`.
8. Перерасчеты:
   - `СреднийЗаработок`, перерасчитывать виды расчета `ОплатаБольничного` и `ДоплатаЗаЗамещение` при изменении видов расчета `ОплатаПоТарифу` и `ДоплатаЗаЗамещение` в следующем периоде.

Анализ задания

Приведенный вариант метаданных не является единственным. Например, введение дополнительного реквизита `ОбъектыОхраны` журнала расчетов вызвано необходимостью формирования отчета по журналу расчетов с детализацией. Есть возможность обойтись и без документа `НачислениеЗарплаты`, делая ввод расчетов непосредственно из документов

ИндивидуальныйНаряд. Но в этом случае получится очень много "похожих" записей, что увеличивает объем хранимой информации.

Поскольку расчет среднего заработка понадобится в двух видах расчета (ОплатаПоБольничному и ДоплатаЗаЗамещение), то вынесем функцию вычисления среднечасового заработка в глобальный модуль (листинг 11.17).

Листинг 11.17. Глобальный модуль

```
Функция глСреднечасовойЗаработок(Объект,ДатаНачала,ДатаОкончания) Экспорт
БазаДляСреднего=ГруппаРасчетов.БазаДляСреднего;
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(РасчетБазы)
|С ДатаНачала по ДатаОкончания;
|Сотрудник = ЖурналРасчетов.Зарплата.Объект;
|ВидРасч = ЖурналРасчетов.Зарплата.ВидРасч;
|Результат = ЖурналРасчетов.Зарплата.Результат;
|Часы = ЖурналРасчетов.Зарплата.Часы;
|Функция База = Сумма(Результат);
|Функция СуммаЧасов = Сумма(Часы);
|Группировка Сотрудник;
|Условие(Сотрудник=Объект);
|Условие(ВидРасч.ВходитВГруппу(БазаДляСреднего)=1);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
        Возврат 0;
КонецЕсли;
Если Запрос.Группировка(1) = 1 Тогда
        Если Запрос.СуммаЧасов>0 Тогда
                Возврат Запрос.База/Запрос.СуммаЧасов;
        Иначе
                Сообщить("За период с "+ДатаНачала+" по "+ДатаОкончания+
                " отработано "+Запрос.СуммаЧасов+" часов");
        КонецЕсли;
КонецЕсли;;
Возврат 0;
КонецФункции;
```

Документ СменаГрафика служит только для привязки графика работы к сотруднику. Привязка выполняется в предопределенной процедуре ОбработкаПроведения() с помощью функции УстановитьРеквизитСправочника() (листинг 11.18). Изменение значения периодического реквизита справочника также является движением документа. Просмотреть движения документа по периодическим реквизитам справочника можно через меню **Операции**, пункт **Движения документа,** в журнале документов. Движения документа удаляются автоматически, если стоит флажок **Автоматическое удаление движений** в форме настройки вида документа. Если же этот флажок не стоит, то движения можно очистить так, как показано в листинге 11.18 в предопределенной процедуре ПриОтменеПроведения().

Листинг 11.18. Модуль документа СменаГрафика

```
Процедура ОбработкаПроведения()
УстановитьРеквизитСправочника(Сотрудник,
        "График", График, ДатаДок);
КонецПроцедуры
Процедура ПриОтменеПроведения()
ОчиститьДвижения("Справочник");
КонецПроцедуры
```

Документ БольничныйЛист содержит сведения о периоде нетрудоспособности сотрудника. При проведении документ формирует запись в журнале расчетов. Если дата начала и дата окончания больничного находятся в разных расчетных периодах, то система автоматически разобьет вводимую запись на несколько. Текст модуля документа БольничныйЛист приведен в листинге 11.19.

Листинг 11.19. Модуль документа БольничныйЛист

```
Процедура ОбработкаПроведения()
ЖР=СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.ВвестиРасчет(Сотрудник,ВидРасчета.ОплатаПоБольничному,
        ДатаНачала,ДатаОкончания);
КонецПроцедуры
```

Документ НачислениеЗарплаты предназначен для ввода записей в журнал расчетов на основании информации о выполненных работах из документов ИндивидуальныйНаряд.

Так как по условию задачи информация вводится еженедельно, а расчет делается раз в месяц, то при проведении документа НачислениеЗарплаты мы

выбираем запросом сведения о том, где и сколько времени отработал сотрудник, и группируем полученную информацию по сотрудникам, объектам и типам работы.

Замечание

Можно было бы дополнительно группировать информацию по документам-нарядам. В этом случае вместо функции `ВвестиРасчет()` можно было бы использовать функцию `ВвестиРасчетНаОсновании()` и указывать ссылку на документ-основание. Это бы упростило (и ускорило!) в дальнейшем расчет введенных записей, но количество самих записей увеличилось бы в 4 раза.

Текст модуля документа `НачислениеЗарплаты` приведен в листинге 11.20.

Листинг 11.20. Модуль документа `НачислениеЗарплаты`

```
Процедура ОбработкаПроведения()
ЖР=СоздатьОбъект("ЖурналРасчетов.Зарплата");
Д1=НачМесяца(Период);
Д2=КонМесяца(Период);
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(ПоНарядам)
|Период С Д1 По Д2;
|Сотрудник = Документ.ИндивидуальныйНаряд.Сотрудник;
|ОбъектОхраны = Документ.ИндивидуальныйНаряд.ОбъектОхраны;
|Часы = Документ.ИндивидуальныйНаряд.Часы;
|СуммаПоТарифу = Документ.ИндивидуальныйНаряд.Сумма;
|ТипРаботы = Документ.ИндивидуальныйНаряд.ТипРаботы;
|Функция ЧасыСумма = Сумма(Часы);
|Функция Сумма = Сумма(СуммаПоТарифу);
|Группировка Сотрудник без групп;
|Группировка ОбъектОхраны без групп;
|Группировка ТипРаботы;
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
       Возврат;
КонецЕсли;
Пока Запрос.Группировка(1) = 1 Цикл
```

```
        Пока Запрос.Группировка(2) = 1 Цикл
            ЖР.УстановитьРеквизит("Часы",Запрос.ЧасыСумма);
            ЖР.УстановитьРеквизит("ОбъектОхраны",Запрос.ОбъектОхраны);
            ЖР.ВвестиРасчет(Запрос.Сотрудник,
                ВидРасчета.ОплатаПоТарифу,Д1,Д2,
                Запрос.Сумма);
            Пока Запрос.Группировка(3) = 1 Цикл
                Если Запрос.ТипРаботы =
                Перечисление.ТипыРаботы.Замещение Тогда
                    ЖР.УстановитьРеквизит("ОбъектОхраны",
                    Запрос.ОбъектОхраны);
                    ЖР.ВвестиРасчет(Запрос.Сотрудник,
                    ВидРасчета.ДоплатаЗаЗамещение,Д1,Д2);
                КонецЕсли;
            КонецЦикла;
        КонецЦикла;
КонецЦикла;
КонецПроцедуры
```

Модули видов расчетов `ОплатаПоТарифу` и `ДоплатаЗаЗамещение`, приведенные в листингах 11.21 и 11.22, схожи тем, что запросом мы выбираем информацию об отработанном времени из документов `ИндивидуальныйНаряд`.

Замечание

Отметим, что запрос по умолчанию выбирает информацию только из проведенных документов, поэтому при перепроведении документов `ИндивидуальныйНаряд` можно сделать нерасчитанными соответствующие записи журнала расчетов.

При записи информации об отработанных часах мы проверяем, не является ли рассчитываемая запись сторнирующей. В этом случае отработанные часы должны быть с минусом. В листинге 11.23 приведен текст модуля расчета оплаты по больничному листу.

Листинг 11.21. Модуль расчета вида расчета `ОплатаПоТарифу`

```
Процедура ПровестиРасчет()
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(ПоНарядам)
|Период С ДатаНачала По ДатаОкончания;
|Сотрудник = Документ.ИндивидуальныйНаряд.Сотрудник;
```

```
|ОбъектОхр = Документ.ИндивидуальныйНаряд.ОбъектОхраны;
|Часы = Документ.ИндивидуальныйНаряд.Часы;
|СуммаПоТарифу = Документ.ИндивидуальныйНаряд.Сумма;
|Функция ЧасыСумма = Сумма(Часы);
|Функция Сумма = Сумма(СуммаПоТарифу);
|Группировка Сотрудник без групп;
|Условие (Сотрудник=Объект);
|Условие (ОбъектОхр=ОбъектОхраны);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
       Возврат;
КонецЕсли;
Часы=?(Сторно=1,-1,1)*Запрос.ЧасыСумма;
Результат=Запрос.Сумма;
КонецПроцедуры
```

Листинг 11.22. Модуль расчета вида расчета `ДоплатаЗаЗамещение`

```
Процедура ПровестиРасчет()
Замещение=Перечисление.ТипыРаботы.Замещение;
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(ПоНарядам)
|Период С ДатаНачала По ДатаОкончания;
|Сотрудник = Документ.ИндивидуальныйНаряд.Сотрудник;
|ОбъектОхр = Документ.ИндивидуальныйНаряд.ОбъектОхраны;
|Часы = Документ.ИндивидуальныйНаряд.Часы;
|ТипРаботы = Документ.ИндивидуальныйНаряд.ТипРаботы;
|Функция ЧасыСумма = Сумма(Часы);
|Группировка Сотрудник без групп;
|Условие (Сотрудник=Объект);
|Условие (ОбъектОхр=ОбъектОхраны);
|Условие (ТипРаботы=Замещение);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
```

```
        Возврат;
КонецЕсли;
// Рассчитываем доплату за замещение
Начало=ДобавитьМесяц(НачМесяца(ДатаНачала),-1);
Конец=ДобавитьМесяц(КонМесяца(ДатаОкончания),-1);
Результат=глСреднечасовойЗаработок(Объект,Начало,Конец)*
Запрос.ЧасыСумма/2;
КонецПроцедуры
```

Листинг 11.23. Модуль расчета вида расчета `ОплатаПоБольничному`

```
Процедура ПровестиРасчет()
// Определяем количество дней на больничном по календарю
Календарь=Объект.График.Получить(ДатаНачала);
Дни=Календарь.Дней(ДатаНачала,ДатаОкончания);
// Рассчитываем оплату по больничному
Начало=ДобавитьМесяц(НачМесяца(ДатаНачала),-1);
Конец=ДобавитьМесяц(КонМесяца(ДатаОкончания),-1);
Результат=глСреднечасовойЗаработок(Объект,Начало,Конец)/2*8*Дни;
КонецПроцедуры
```

Пишем отчет

Отчет должен иметь следующий вид:

Работник/Объект	**Август 2005**		**Сентябрь 2005**	
	Проработано часов	**Сумма заработка**	**Проработано часов**	**Сумма заработка**
Иванов И.И., в т. ч.	*A*			
Магазин "Весна"	*B*			
Магазин "Одежда"	*B*			
Петров П.П., в т. ч.	*A*			
Магазин "Весна"	*B*			
Магазин "Одежда"	*B*			
Итого	*C*			

Очевидно, что информация должна извлекаться из журнала расчетов и иметь три группировки: `Сотрудник`, `ОбъектОхраны` и `ПериодДействия`. Как определить, в какой последовательности?

В таблице имеются ячейки трех типов:

- *A* — итог по группировкам `Сотрудник` и `ПериодДействия`;
- *B* — итог по группировкам `Сотрудник`, `ОбъектОхраны` и `ПериодДействия`;
- *C* — итог по группировке `ПериодДействия`.

Чтобы получить итоги всех типов, мы должны выбрать следующий порядок группировок: `ПериодДействия`, `Сотрудник`, `ОбъектОхраны`.

Однако после выполнения запроса информация будет представлена в следующем виде:

ПериодДействия	Сотрудник	ОбъектОхраны	Часов	Сумма
*	*	*	1000	100 000
январь	*	*	300 (*C*)	30 000
январь	Иванов	*	100 (*A*)	10 000
январь	Иванов	Магазин1	50 (*B*)	5000
январь	Иванов	Магазин2	50 (*B*)	5000
январь	Петров	*	200 (*A*)	20 000
январь	Петров	Магазин1	100 (*B*)	10 000
январь	Петров	Магазин2	100 (*B*)	10 000
февраль	*	*	400 (*C*)	40 000
...				

Чтобы переупорядочить результат запроса, выгрузим его в таблицу значений и выполним сортировку:

```
Запрос.Выгрузить(ТЗ,0,1);
ТЗ.Сортировать("Сотрудник,ОбъектОхраны,ПериодДействия");
```

Совет

Для удобства просмотра таблицу значений `ТЗ` лучше создать в форме диалога. Тогда вы можете посмотреть содержимое таблицы.

Теперь у нас есть все необходимые данные в удобном для вывода в отчет виде. Перебирая строки из таблицы значений, мы начинаем новую строку отчета командой `ВывестиСекцию()`, когда в текущей строке появляется новый сотрудник или новый объект охраны. В противном случае, мы

присоединяем ячейки слева — это информация по периодам действия. Так как по сотруднику или объекту в таблице значений могут быть представлены не все периоды, то по отсутствующим периодам мы выводим пустые ячейки. Текст отчета приведен в листинге 11.24, а шаблон отчета — на рис. 11.15.

Листинг 11.24. Отчет по расчетным данным

```
Процедура Сформировать()
ПервыйПериод=НачМесяца(ВыбНачПериода);
ПоследнийПериод=НачМесяца(ВыбКонПериода);
// Запрос по расчетным данным
ОплатаПоБольничному=ВидРасчета.ОплатаПоБольничному;
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(Сформировать)
|С ВыбНачПериода по ВыбКонПериода;
|Сотрудник = ЖурналРасчетов.Зарплата.Объект;
|ВидРасч = ЖурналРасчетов.Зарплата.ВидРасч;
|ОбъектОхраны = ЖурналРасчетов.Зарплата.ОбъектОхраны;
|ПериодДействия = ЖурналРасчетов.Зарплата.ПериодДействия.ДатаНачала;
|Результат = ЖурналРасчетов.Зарплата.Результат;
|Часы = ЖурналРасчетов.Зарплата.Часы;
|Функция СуммаЗаработка = Сумма(Результат);
|Функция ПроработаноЧасов = Сумма(Часы);
|Группировка ПериодДействия;
|Группировка Сотрудник;
|Группировка ОбъектОхраны;
|Условие (ВидРасч<>ОплатаПоБольничному);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
       Возврат;
КонецЕсли;
ТЗ=СоздатьОбъект("ТаблицаЗначений");
Запрос.Выгрузить(ТЗ,0,1);
// Упорядочим данные так, как нужно в отчете
ТЗ.Сортировать("Сотрудник,ОбъектОхраны,ПериодДействия");
```

```
Таб=СоздатьОбъект("Таблица");
Таб.ВывестиСекцию("Шапка");
// Выводим заголовок
Таб.ВывестиСекцию("Заголовок|Начало");
ЧислоСтолбцов=1;
Период=ПервыйПериод;
Пока Период<=ПоследнийПериод Цикл
        ПечПериод=""+ДатаМесяц(Период)+"."+ДатаГод(Период);
        Таб.ПрисоединитьСекцию("Заголовок|Период");
        Период=ДобавитьМесяц(Период,1);
        ЧислоСтолбцов=ЧислоСтолбцов+2;
КонецЦикла;
// Выводим данные по сотрудникам и объектам
ТекСотрудник=ПолучитьПустоеЗначение("Справочник.Сотрудники");
ТекОбъект=ПолучитьПустоеЗначение("Справочник.ОбъектыОхраны");
ТекПериод=ПолучитьПустоеЗначение("Дата");
ТекСекция="Сотрудник";
ЧислоСтрок=0;
ТЗ.ВыбратьСтроки();
Пока ТЗ.ПолучитьСтроку()=1 Цикл
        Если ТЗ.Сотрудник.Выбран()=0 Тогда
                // Итоги по месяцам выводим после цикла
                Продолжить;
        КонецЕсли;
        ПечСотрудник=ТЗ.Сотрудник;
        ПечОбъект=ТЗ.ОбъектОхраны;
        ПечПериод=ТЗ.ПериодДействия;
        ПечЧасов=ТЗ.ПроработаноЧасов;
        ПечСумма=ТЗ.СуммаЗаработка;
        Если ПечСотрудник<>ТекСотрудник Тогда
                // Началась новая строка по сотруднику
                ТекСекция="Сотрудник";
                Таб.ВывестиСекцию(ТекСекция+"|Начало");
                ЧислоСтрок=ЧислоСтрок+1;
                ТекСотрудник=ПечСотрудник;
                ТекПериод=ПервыйПериод;
        ИначеЕсли (ПечОбъект.Выбран()=1) и (ПечОбъект<>ТекОбъект) Тогда
                // Началась новая строка по объекту
```

```
                ТекСекция="ОбъектОхраны";
                Таб.ВывестиСекцию(ТекСекция+"|Начало");
                ЧислоСтрок=ЧислоСтрок+1;
                ТекОбъект=ПечОбъект;
                ТекПериод=ПервыйПериод;
        КонецЕсли;
        // Если есть пропуск данных по некоторому периоду, заполняем его
        // пустыми ячейками
        Пока ТекПериод<ПечПериод Цикл
                Таб.ПрисоединитьСекцию(ТекСекция+"|Пусто");
                ТекПериод=ДобавитьМесяц(ТекПериод,1);
        КонецЦикла;
        // Выводим информацию о сумме и часах
        Таб.ПрисоединитьСекцию(ТекСекция+"|Период");
        ТекПериод=ДобавитьМесяц(ПечПериод,1);
КонецЦикла;
// Выводим итоги по месяцам
Таб.ВывестиСекцию("Итого|Начало");
Период=ПервыйПериод;
Пока Период<=ПоследнийПериод Цикл
        Нашли=0;
        Запрос.ВНачалоВыборки();
        Пока Запрос.Группировка(1)=1 Цикл
                Если Запрос.ПериодДействия=Период Тогда
                        Нашли=1;
                        ПечЧасов=Запрос.ПроработаноЧасов;
                        ПечСумма=Запрос.СуммаЗаработка;
                        Таб.ПрисоединитьСекцию("Итого|Период");
                КонецЕсли;
        КонецЦикла;
        Если Нашли=0 Тогда
                Таб.ПрисоединитьСекцию("Итого|Пусто");
        КонецЕсли;
        Период=ДобавитьМесяц(Период,1);
КонецЦикла;
// Прорисовываем рамку
Таб.Область(6,1,6+ЧислоСтрок,ЧислоСтолбцов).Рамка(3,3,3,3);
Таб.Показать("Сформировать", "");
КонецПроцедуры
```

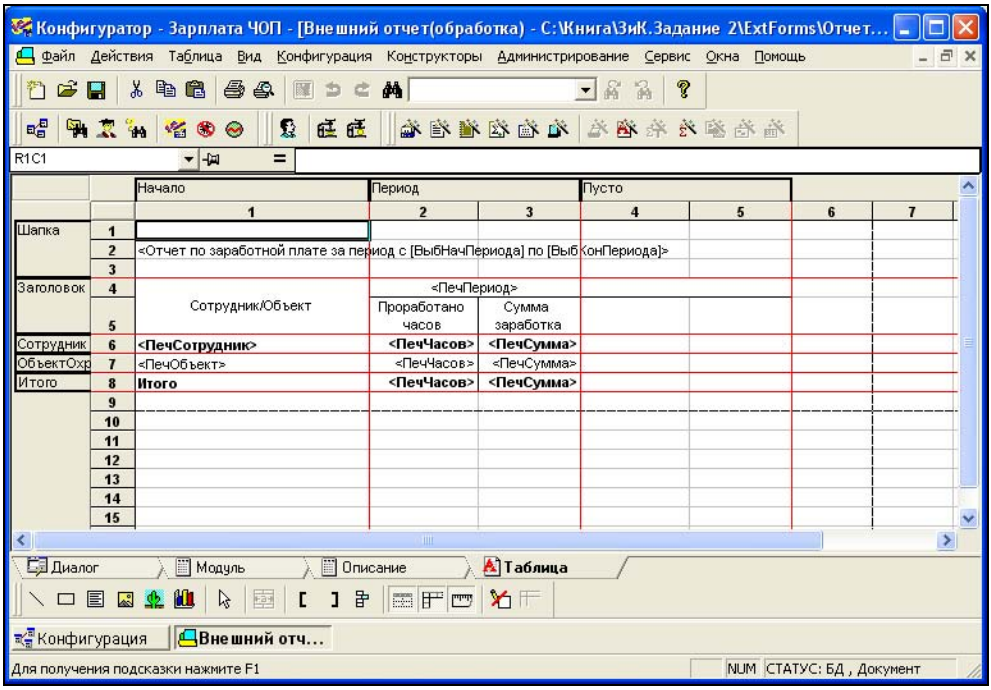


Рис. 11.15. Шаблон печатной формы отчета

11.4. Полезные советы при работе с программой "Зарплата и Кадры"

Наиболее часто с компонентой "Расчет" приходится встречаться при настройке и сопровождении конфигурации "Зарплата и Кадры". Отметим ее особенности.

11.4.1. Способы начисления зарплаты

В программе есть два основных способа начисления зарплаты.

Первый способ — расчет методом отклонений (система оплаты: простая повременная по окладу по дням, простая повременная оплата по окладу по часам и др.). В этом случае в приказе о приеме на работу и далее при кадровых перемещениях указывается оклад и график работы сотрудника. При начислении зарплаты считается, что сотрудник отработал все отведенное по графику время за исключением отклонений. Отклонения вводятся специальными документами, такими как "Больничный лист", "Отпуск", "Невыход" и др. Табель отработанного времени в этом случае формируется как отчет.

Замечание

Алгоритмы расчетов больничного листа, отпускных, налога на доходы, единого социального налога достаточно жестко регламентированы, и изменения в расчетах по ним происходят в законодательстве практически ежегодно. Фирма "1С" достаточно быстро отслеживает эти изменения в своей программе и выпускает новые релизы.

Второй способ — оплата по табелю отработанного времени. В этом случае вводится документ "Табель отработанного времени", в котором фиксируется фактическое время, отработанное сотрудником.

11.4.2. Пользовательские виды расчета

Так как расчет заработной платы, начисление налогов и кадровый учет достаточно жестко регламентированы, то основные виды расчетов "зашиты" внутри конфигурации. А как быть с теми видами расчетов, которые пользователь хочет добавить сам? В этом смысле конфигурация спроектирована достаточно универсально и не требует вызова программиста в наиболее простых случаях. Как это сделать, ведь новые виды расчета можно добавлять только в конфигураторе? Разработчики вышли из положения следующим образом: в конфигурации были созданы виды расчета с названиями "ПроизвольнаяДоплата01", "ПроизвольнаяДоплата02" и т. д. (всего 80); "ПроизвольноеУдержание01", "ПроизвольноеУдержание02" и т. д. (всего 30) и другие произвольные виды расчета. Для работы с произвольными видами расчтетов и настройки предопределенных видов расчета создан справочник ВидыРасчетов, в котором, в частности, определяется алгоритм, по которому производится вычисление результата произвольного вида расчета. Возможные следующие алгоритмы:

- процентом от базы;
- суммой пропорционально отработанным дням;
- по тарифу;
- фиксированной суммой;
- процентом от минимальной месячной оплаты труда;
- по разряду единой тарифной сетки;
- суммой за отработанный день;
- суммой пропорционально отработанным часам.

База для произвольных видов расчета и корректировка базы для предопределенных видов расчета выполняются в подчиненном справочнике ВидыРасчетовБаза.

11.4.3. Формирование проводок по расчетным данным и налоговые отчеты

Конфигурация "Зарплата и Кадры" использует исключительно базовые объекты и объекты компоненты "Расчет". Поэтому проводки как таковые в программе не хранятся, а формируются в виде отчета по расчетным данным, хранящимся в журналах расчетов, и настройкам шаблонов проводок. План счетов, виды и значения субконто хранятся в справочниках.

Шаблоны проводок имеются у многих объектов:

- константа "ПроводкаПоУмолчанию";
- справочник "ВидыРасчетов";
- справочник "ДополнительныеПроводки";
- справочник "Подразделения";
- справочник "Сотрудники";
- некоторые виды документов.

Совет

Чтобы найти, какие объекты метаданных имеют ссылки на интересующий объект (в данном случае, это `Справочник.ШаблоныПроводок`), нужно подвести к нему курсор мыши в дереве метаданных, нажать правую кнопку и выбрать пункт **Поиск ссылок на объект**.

При формировании проводки по записи журнала расчетов производится подстановка шаблона по принципу "от частного к общему", то есть сначала смотрим, есть ли шаблон проводки у документа, потом у сотрудника, затем у подразделения, в котором он работает, у вида расчета и, наконец, обращаемся к проводке по умолчанию, заданной в константе.

Проводки формируются по периоду регистрации, при этом может задействоваться счет учета будущих периодов (для хозрасчетных организаций), который автоматически распределяется по счетам затрат в последующих периодах.

При достаточно большом числе сотрудников (свыше тысячи) формирование проводок за месяц может выполняться несколько часов.

Налоговые отчеты в конфигурации "Зарплата и Кадры" формируются на основании расчетных данных, независимо от того, настроены шаблоны проводок или нет. Расчет налогов производится в отдельном журнале расчетов по каждому сотруднику.

11.4.4. Выгрузка проводок в "1С:Бухгалтерию 7.7"

Для настройки корректного обмена данными с программой "1С:Бухгалтерия 7.7" необходимо:

1. Выгрузить план счетов и виды субконто со значениями из бухгалтерии.

2. Загрузить план счетов и виды субконто в зарплату.
3. Настроить шаблоны проводок.
4. В случае изменения плана счетов или справочников в бухгалтерии нужно повторить пункты 1—3.
5. После этого ежемесячно производится выгрузка проводок из зарплаты и их загрузка в бухгалтерию. Проводки загружаются в виде ручной операции.

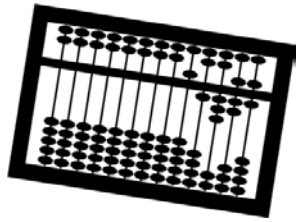
11.4.5. Смена периода расчета зарплаты

Смена периода расчета зарплаты производится ежемесячно, периода расчета налогов — ежегодно. При попытке смены периода расчета "назад" система обнуляет все результаты записей журнала расчетов (в том числе и ручные), поэтому иногда, в некритичных для расчета случаях, можно вернуться назад с помощью обработки, приведенной в листинге 11.7.

11.5. Контрольные вопросы

1. Для решения каких задач предназначена компонента "Расчет"? Приведите несколько примеров задач, не связанных с расчетом заработной платы, для которых можно применить "Расчет". Какие объекты метаданных являются специфическими для "Расчета"?
2. Каково назначение календарей в системе? Основные принципы их создания и методы работы с календарями. Что нужно учитывать при выборе стартовой даты календаря?
3. Какие действия с календарями доступны пользователям? Какие недоступны? Обязательно ли использование операции автоматического заполнения календаря? Могут ли календари редактироваться с помощью встроенного языка?
4. Понятие объекта "Праздники". Для чего он нужен и обязательно ли его использование? Какие методы доступны при работе с объектом? Можно ли создать несколько объектов типа "Праздники"?
5. Каково назначение видов расчета? Что такое "Приоритет вида расчета"? Что необходимо учитывать при выборе приоритета вида расчета?
6. Каково основное назначение групп расчетов? Методы работы с группами расчетов. Где задается вхождение вида расчетов в группу?
7. Что такое "Вытесняющий расчет"? На чем основан механизм автоматического сторнирования записей журнала расчетов?
8. Для чего служат журналы расчетов? Сколько журналов расчетов может быть создано в рамках одной конфигурации? Какими данными характеризуются строки журнала расчетов?

9. Какими свойствами обладают журналы расчетов? Какие из них являются важнейшими? Какие реквизиты журналов расчетов являются обязательными?
10. Какое количество дополнительных реквизитов может быть создано в журналах расчетов? Какими способами в них может быть внесена информация?
11. Какую периодичность может иметь журнал расчетов? Можно ли изменить периодичность журнала расчетов, если он содержит записи? Ваши действия, если требуемая по условиям задачи периодичность не предусмотрена программой (например, 10 дней)?
12. Что происходит при смене расчетного периода? Можно ли изменить расчетный период на несколько периодов вперед? На несколько периодов назад? Что произойдет в этих случаях? Можно ли изменить период журнала расчетов средствами встроенного языка?
13. Какие средства существуют для добавления записей в журнал расчетов? Для удаления записей?
14. Может ли быть переопределен программными средствами атрибут "Сторно" журнала расчетов? В каких случаях?
15. Понятие "Правил перерасчета". Каково их назначение? Какие виды расчета с точки зрения "Правил перерасчета" считаются ведущими? Какие зависимыми?
16. Что происходит в программе при наличии "Правила перерасчета" и манипулировании в журнале расчетов с записью, содержащей ведущий вид расчета? Могут ли установленные при конфигурировании правила перерасчета отключаться или переназначаться программными средствами?



Глава 12

Операции экспорта-импорта данных

На практике очень часто возникает необходимость в осуществлении переноса данных из одной программы в другую. Система 1С:Предприятие предоставляет мощные механизмы для выполнения таких операций.

12.1. Работа с файлами

Для работы с файлами в системе используется специальный агрегатный тип данных — `ФС`. По умолчанию в системе всегда доступен уже существующий объект с именем `ФС`, к которому можно применять методы объекта типа `"ФС"`. Кроме того, можно создать произвольное число объектов типа `"ФС"` с помощью функции `СоздатьОбъект("ФС")`.

У объекта типа `"ФС"` имеются стандартные функции для работы с файлами:

- `ВыбратьФайл()` — открывает диалог выбора файла;
- `ВыбратьКаталог()` — открывает диалог выбора каталога;
- `СуществуетФайл(<ИмяФайла>)` — проверяет, существует ли файл с указанным именем;
- `КопироватьФайл(<ИмяФайлаИсточника>,<ИмяФайлаПриемника>)` — копирует файл-источник в файл-приемник;
- `УдалитьФайл(<ИмяФайла>)` — удаляет файл;
- `ПереименоватьФайл()` и др.

12.2. Использование текстовых файлов для переноса данных

Для работы с текстами в системе используется специальный тип данных `"Текст"`. Средства языка позволяют выводить строки в текстовые файлы и

считывать из имеющихся файлов текст с последующим разбором его по строкам.

Внимание!

Тип данных "Текст" предназначен для работы с небольшими (размером до 64 Кбайт) файлами. Для работы с большими текстами нужно подключать внешнюю компоненту, например V7Plus.dll, в которой есть объект для работы с текстовыми файлами V7TextFile.

При работе с объектом типа "Текст" используются функции:

- Открыть(<ИмяФайла>) — открывает файл;
- КодоваяСтраница(<Режим>) — получить/установить режим кодировки. <Режим> =: 0 — Windows-кодировка, 1 — DOS-кодировка;
- Показать(<Заголовок>,<ИмяФайла>) — открыть окно редактирования текста;
- Записать(<ИмяФайла>) — записывает текст в файл с указанным именем.

12.2.1. Чтение текста

При чтении текста используются функции:

- КоличествоСтрок() — количество строк в тексте;
- ПолучитьСтроку(<НомерСтроки>) — получить строку текста по номеру.

12.2.2. Запись текста

При записи текста используются функции:

- ВставитьСтроку(<НомерСтроки>,<Строка>) — вставить строку с указанным номером;
- ЗаменитьСтроку(<НомерСтроки>,<Строка>) — заменить строку с указанным номером;
- УдалитьСтроку(<НомерСтроки>) — удалить строку с указанным номером;
- ДобавитьСтроку(<Строка>) — добавить строку в конец текста;
- Очистить() — удалить все строки текста.

В качестве примера рассмотрим следующую задачу.

Необходимо из одной информационной базы выгрузить справочник номенклатуры и загрузить его в аналогичный справочник другой информационной базы. Запись элементов справочника в текстовый файл показана в листинге 12.1, а чтение элементов из файла — в листинге 12.2.

Листинг 12.1. В файл exp_imp.txt выгружается справочник номенклатуры

```
Текст = СоздатьОбъект("Текст");
Список = СоздатьОбъект("СписокЗначений");
Спр=СоздатьОбъект("Справочник.Номенклатура");
Спр.ВыбратьЭлементы();
Пока Спр.ПолучитьЭлемент()=1 Цикл
        Список.ДобавитьЗначение(Спр.Код);
        Список.ДобавитьЗначение(Спр.Наименование);
        // И другие поля…
        // Выводим список в текст
        Текст.ДобавитьСтроку(Список.ВСтрокуСРазделителями());
        Список.УдалитьВсе();// Очищаем список
КонецЦикла;
Текст.Записать("exp_imp.txt");
```

Листинг 12.2. Из файла exp_imp.txt загружается справочник номенклатуры

```
Спр=СоздатьОбъект("Справочник.Номенклатура");
Текст = СоздатьОбъект("Текст");
Список = СоздатьОбъект("СписокЗначений");
Текст.Открыть("exp_imp.txt");
Для Ном=1 По Текст.КоличествоСтрок() Цикл
Стр = Текст.ПолучитьСтроку(Ном);
Список.ИзСтрокиСРазделителями(Стр); // Преобразуем строку
// в список значений
Если Спр.НайтиПоКоду(Список.ПолучитьЗначение(1))=0 Тогда
        Спр.Новый();
        Спр.Код= Список.ПолучитьЗначение(1);
        Спр.Наименование= Список.ПолучитьЗначение(2);
Спр.Записать();
КонецЕсли;
КонецЦикла;
```

12.3. Работа с файлами в формате DBF

Для работы с базами данных в формате DBF используется объект типа "Xbase". При этом надо учитывать следующие ограничения:

- не поддерживаются поля типа memo;

- ❒ база данных открывается монопольно;
- ❒ поддерживаются только индексные файлы в формате CDX.

Доступ к полям базы данных осуществляется записью через точку аббревиатуры `БД` и имени поля, например, чтобы получить значения поля `NAME` базы данных, открытой в переменной `БД`, нужно написать `БД.NAME`.

В листинге 12.3 приведен способ чтения всех записей базы данных, хранящейся в файле data.dbf и имеющей два поля: CODE и NAME.

Листинг 12.3. Прочтение всех записей базы данных, имеющей два поля

```
БД=СоздатьОбъект("XBase");
БД.ОткрытьФайл("data.dbf");
// Проверяем, удалось ли открыть базу данных
Если БД.Открыта()=1 Тогда
// Встаем на первую запись
Если БД.Первая()=1 Тогда
Пока 1=1 Цикл
        // Обработка записи базы данных
        Сообщить(Строка(БД.CODE) + " " + БД.NAME);
        Если БД.Следующая()=0 Тогда
                // Не удалось перейти к следующей записи
                Прервать;
        КонецЕсли;
КонецЦикла;
КонецЕсли;
КонецЕсли;
```

12.4. Обмен данными с помощью OLE Automation

Для запуска системы 1С:Предприятие в качестве OLE Automation сервера из внешнего приложения (например, из другой программы 1С) выполняется следующая последовательность действий:

1. Создается объект с OLE-идентификатором:
   - V77.Application;
   - V77S.Application — SQL версия системы 1С:Предприятие 7.7;
   - V77L.Application — локальная версия системы 1С:Предприятие 7.7;
   - V77M.Application — сетевая версия системы 1С:Предприятие 7.7.

2. Выполняется инициализация системы 1С:Предприятие 7.7 методом Initialize.
3. Вызываются атрибуты и методы системы 1С:Предприятие 7.7 как OLE Automation сервера.

Система 1С:Предприятие в качестве OLE Automation сервера предоставляет полный доступ к своему *глобальному контексту*, а также имеет 4 метода:

1. `Initialize()` — выполняет инициализацию системы 1С:Предприятие 7.7.
2. `CreateObject()` — создает объект агрегатного типа данных 1С:Предприятия 7.7 и возвращает ссылку на него.
3. `EvalExpr()` — вычисляет выражение системы 1С:Предприятие 7.7.
4. `ExecuteBatch()` — выполняет последовательность операторов системы 1С:Предприятие 7.7.

Замечание

Информационная база открывается в монопольном режиме.

В листинге 12.4 приведен пример открытия внешней информационной базы, расположенной в каталоге `Путь`, под именем пользователя `Пользователь`. В этой внешней базе создается объект типа `"Справочник.Номенклатура"`, и в окно сообщений выводится список всех наименований элементов справочника номенклатуры.

Листинг 12.4. Обработка элементов справочника другой информационной базы в режиме OLE Automation

```
V7 = СоздатьОбъект("V77.Application");
Открыта = V7.Initialize(V7.RMTrade, "/d" + Путь +
" /M /N" + Пользователь, "");
Если Открыта = 0 Тогда
        Предупреждение("Ошибка открытия информационной базы");
        Возврат;
КонецЕсли;
// Создаем объект во внешней базе
ВнешнийСпр = V7.CreateObject("Справочник.Номенклатура");
// Перебираем элементы
ВнешнийСпр.ВыбратьЭлементы();
Пока ВнешнийСпр.ПолучитьЭлемент() = 1 Цикл
        Сообщить(ВнешнийСпр.Наименование);
КонецЦикла;
```

Достоинство метода OLE Automation в том, что пишется фактически одна процедура загрузки данных, без предварительной выгрузки. А недостаток в том, что при использовании этого метода необходим монопольный доступ к информационной базе на этом же компьютере или в локальной сети, откуда производится загрузка данных.

Задание 12.1

Необходимо выполнить загрузку в типовую конфигурацию "Бухгалтерский учет" справочника сотрудников из файла data.dbf, подготовленного с помощью программы MS Excel.

Kod	Sotr	Oklad	Dolgnost	Podr
1	Иванов	1000	директор	дирекция
2	Петров	2000	бухгалтер	бухгалтерия
3	Романов	3000	менеджер	отдел продаж
4	Сидоров	4000	сторож	автостоянка

При загрузке необходимо:

1. Учитывать возможность наличия в информационной базе сотрудника с таким же кодом.
2. Значения периодических реквизитов должны переноситься на начало текущего квартала.
3. Для реквизитов "должность" и "подразделение" должны создаваться соответствующие элементы справочников.
4. Сформировать отчет о загруженных сотрудниках (формируется одновременно с загрузкой):

Отчет об импортированных элементах справочника "Сотрудники"

Код	ФИО	Оклад	Должность	Подразделение	Статус
1	Иванов	10 000	директор	дирекция	добавлен
2	Петров	8000	бухгалтер	бухгалтерия	пропущен
3	Романов	6000	менеджер	отдел продаж	добавлен
4	Сидоров	4000	сторож	автостоянка	добавлен

Статус принимает значения: *добавлен* или *пропущен*.

Заключение

Итак, уважаемый читатель, вы добрались до заключения. Надеюсь, что чтение не было для вас слишком утомительным, и вы полны сил продолжить дальнейшее изучение системы 1С:Предприятие 7.7. Попробуем дать рекомендации.

Документация к программе 1С:Предприятие 7.7

Документация, как правило, состоит из двух томов по конфигурированию и администрированию, двух томов по описанию встроенного языка, небольшой брошюрки по инсталляции системы и описанию типовых конфигураций, входящих в комплект.

Документация написана вполне добротно, снабжена достаточным числом примеров и пояснений. Естественно документация не заменяет учебник, а учебник не заменяет документацию, так как они имеют разное назначение. Цель документации описать все возможности системы в деталях, а в учебнике материал дается выборочно и последовательно от простых понятий к более сложным.

Типовые конфигурации системы 1С:Предприятие 7.7

Типовые конфигурации включают в себя десятки тысяч строк кода, написанного на встроенном языке. С одной стороны, это опыт практического написания самых тиражируемых конфигураций, с другой стороны — знание этих конфигураций изнутри поможет разъяснять пользователю некоторые особенности их функционирования, не описанные в документации.

Диски информационно-технологического сопровождения (ИТС)

На этих дисках публикуется очень много информации "из первых уст" — от производителя системы. Оформить подписку на диск (он выпускается ежемесячно) можно через любого партнера фирмы "1С". Перечислю некоторые из тем методической поддержки системы 1С:Предприятие 7.7:

- вопросы производительности конфигураций;
- вопросы архитектуры 1С:Предприятия;
- технологические вопросы;
- особенности работы с базами в формате SQL;
- вопросы технологии создания внешних компонент.

И множество других тем.

Сайт фирмы "1С" www.1c.ru

На этом сайте публикуется оперативная информация о выходе новых продуктов, различных мероприятиях (семинарах, учебных курсах, выездных аттестациях и т. п.). В разделе "Обучение" есть график проведения занятий в учебных центрах № 1, 2, 3. В разделе учебного центра № 1 есть примеры задач, которые выдаются на аттестационных экзаменах по системе 1С:Предприятие 7.7.

Особо хочется отметить закрытую конференцию по системе 1С:Предприятие. Чтобы попасть на нее, достаточно иметь аттестат "1С:Профессионал" по любой из компонент 1С:Предприятия. Экзамен "1С:Профессионал" проходит в виде электронного теста из 14 вопросов. Для сдачи экзамена необходимо правильно ответить не менее чем на 12 вопросов. Экзамен можно сдать в любом авторизованном учебном центре, список которых приведен на сайте.

В закрытой конференции можно оперативно получить ответы от опытных специалистов фирм-партнеров фирмы "1С" на технические вопросы по работе с системой 1С:Предприятие и основными типовыми конфигурациями.

Другие информационные ресурсы, посвященные системе 1С:Предприятие

Отметим ряд других сайтов, которые могут быть полезны при практической работе с системой 1С:Предприятие.

Адрес сайта	Название сайта	Авторы, разработчики
http://www.mista.ru	Учебник для начинающих	Станислав Митичкин
http://www.1c-school.ru/	Школа 1С	Алексей Колосов, Владислав Аврутин
http://www.vitalikk.ru	1С:Предприятие 7.7 (Шаг за шагом)	Виталий Кожевников
http://www.gendin.ru/	База знаний по md-файлам	Дмитрий Гендин
http://1c.proclub.ru/	Клуб профессионалов 1С	Илья Лумпов

Приложение

Ответы и решения

В этом приложении приведены ответы и решения на задачи и вопросы, обычно вызывающие затруднения.

Глава 1

Задания

1.1. Если вы правильно установили пароль на конфигурацию, то этот пароль будет запрашиваться при открытии конфигурации, загрузке измененной конфигурации, объединении конфигураций и при запуске отладки конфигурации. Дополнительно можно установить пароль на внешний отчет (обработку), причем можно сделать так, чтобы этот пароль запрашивался и перед выполнением отчета или обработки.

1.2. Интерфейс кассира будет предоставлять доступ через меню и панель инструментов к кассовым документам: "ПриходныйОрдер", "РасходныйОрдер", а также к журналу "Касса". При настройке прав кассира для корректной работы необходимо дать ему возможность чтения или использования большинства объектов (табл. П1). В самом деле, при запуске "1С:Бухгалтерии" открывается обработка "Путеводитель", поэтому если не дать права на использование обработок, то путеводитель не откроется, а будет выдано сообщение "Недостаточно прав доступа!". Если не дать прав на чтение плана счетов, то кассир не сможет выбрать корреспондирующий счет в приходном или расходном ордере.

Если на большинство объектов права можно давать целиком на все объекты данного типа, то на документы надо давать права выборочно. Отметим, что виды прав на документ зависят от того, каким компонентам он принадлежит. Открываем права на документ "ПриходныйОрдер" и даем права на чтение, ввод нового, корректировку, выбор, проведение документа. На остальные действия прав не даем. Аналогично поступаем с правами на документ "РасходныйОрдер".

Таблица П1. *Настройка прав на объекты*

Операция	Пояснение
Конфигурация	
Административные функции	Доступ к функциям Конфигуратора: изменение структуры метаданных, управление списком пользователей, назначение прав доступа, редактирование пользовательских интерфейсов
Сохранение/Выгрузка данных	Создание архивной копии и выгрузка данных в файл переноса данных
Управление оперативными итогами	Смена точки актуальности итогов и открытие нового периода
Управление бухгалтерскими итогами	Расчет итогов
Монитор	Работа с монитором пользователей
Использование в качестве OLE Automation сервера	Доступ к данным системы 1С:Предприятие из внешних программ
Удаление помеченных объектов	Выполнение операции физического удаления данных из информационной базы
Поиск ссылок на объект	Поиск перекрестных ссылок между объектами данных
Использование любых внешних отчетов и обработок	Использование внешних отчетов (обработок)
Использование общих внешних отчетов и обработок	Использование внешних отчетов (обработок), расположенных в подкаталоге ExtForms каталога информационной базы
Использование функций в табло и формульном калькуляторе	Использование в табло и формульном калькуляторе функций встроенного языка системы 1С:Предприятие
Групповое проведение документов	Использование режима группового проведения документов и восстановления последовательностей документов
Использование табло счетов	Использование табло счетов
Монопольный запуск	Запуск системы 1С:Предприятие в монопольном режиме
Автообмен распределенной ИБ	Работа с режимом автообмена

Таблица П1 *(продолжение)*

Операция	Пояснение
Константы	
Чтение	Просмотр значения константы
Корректировка	Изменение значения константы
Справочники	
Чтение	Просмотр справочника
Любые изменения	Все операции
Ввод нового	Ввод новых элементов и групп справочника
Удаление	Удаление элементов и групп справочника
Пометка на удаление	Пометка элементов и групп справочника на удаление
Снятие пометки на удаление	Снятие пометки на удаление с элементов и групп справочника
Корректировка	Редактирование элементов и групп справочника
Документы	
Чтение	Просмотр документов
Любые изменения	Все операции
Ввод нового	Ввод нового документа
Удаление	Удаление документа
Пометка на удаление	Пометка документов на удаление
Снятие пометки на удаление	Снятие пометки на удаление с документов
Корректировка	Редактирование документа
Выбор	Выбор документа
Просмотр подчиненных документов	Просмотр документов, имеющих в основе текущий документ
Проведение документа	Проведение документов
Изменения проведенных документов	Редактирование документов после проведения

Таблица П1 *(продолжение)*

Операция	Пояснение
Изменение документов без перепроведения	Редактирование проведенных документов без повторного проведения
Проведение документов "задним числом"	Проведение документов до точки актуальности итогов
Редактирование архивного документа расчета	Редактирование архивного документа расчета
Проведение архивного документа расчета	Проведение архивного документа расчета
Редактирование операции документа	Редактирование бухгалтерской операции, сформированной документом
Журналы	
Чтение	Просмотр журнала
Просмотр полного журнала	Просмотр полного журнала
Просмотр общего журнала	Просмотр общего журнала
Отчеты, обработки	
Использование	Формирование отчета
Планы счетов	
Чтение	Просмотр планов счетов
Любые изменения	Все операции
Ввод нового	Ввод нового счета
Удаление	Удаление счета
Пометка на удаление	Пометка счетов на удаление
Снятие пометки на удаление	Снятие пометки на удаление со счетов
Корректировка	Редактирование характеристик счета
Операция	
Чтение	Просмотр операций
Просмотр списка	Просмотр журнала операций
Ввод операции без проверки проводок	Возможность отключения режима проверки корректности проводок
Включение проводок операции	Сделать проводки операции активными

Таблица П1 *(окончание)*

Операция	Пояснение
Выключение проводок операции	Сделать проводки операции неактивными
Проводки	
Просмотр списка	Просмотр журнала проводок
Просмотр списка корректных проводок	Возможность присмотра списка корректных проводок
Редактирование списка корректных проводок	Возможность редактирования списка корректных проводок
Регистры	
Чтение	Просмотр движений регистров
Журналы расчетов	
Чтение	Просмотр журнала расчетов
Любые изменения	Редактирование журнала расчетов
Расчет	Выполнение расчета записей журнала расчетов
Изменение расчетного периода	Смена расчетного периода журнала расчетов
Календари	
Чтение	Просмотр календаря
Любые изменения	Редактирование календаря
Редактировать праздники	Редактирование календаря праздничных дней

Контрольные вопросы

Информационная база может храниться в формате DBF и в базе данных под управлением MS SQL Server.

Одна и та же конфигурация может работать и в локальной, и в сетевой версии.

Ключи красного цвета используются для сетевых версий и УРИБ, для локальных версий — ключи белого цвета.

Для того чтобы запустить 1С:Предприятие 7.7 без открытия диалога выбора информационной базы, нужно указать в строке запуска программы ключ `/d`

и каталог информационной базы. Если имя каталога содержит пробелы, то его нужно заключить в кавычки.

Тестирование и исправление информационной базы выполняется для устранения ошибок в файлах базы данных. Отметим, что при серьезном нарушении файлов эта процедура может не помочь. Тогда единственный способ — восстановить полностью или частично файлы из архивной копии.

Если удалить файл users.usr, то список пользователей будет уничтожен, и вход в систему будет выполняться под интерфейсом, у которого стоит признак "Использовать без авторизации".

Чтобы узнать, какие изменения были сделаны в конфигурации, нужно выполнить объединение измененной конфигурации с неизмененной и сформировать подробный отчет. Естественно, сохранять объединенную базу не нужно, так как нам необходим только отчет.

Глава 2

Решения приведены для 468-го релиза типовой конфигурации "Бухгалтерский учет. Редакция 4.5". Для других релизов план счетов и аналитика по субконто может отличаться.

Задания

2.1. Решение приведено на рис. П1. Оборотно-сальдовая ведомость по этой операции показана на рис. П2. Чтобы получить такой вид оборотно-сальдовой ведомости, нужно убрать флажок **Данные по субсчетам и субконто** и **Данные по забалансовым счетам**.

2.2. Проводки, введенные вручную и сделанные документами, могут отличаться на суммы налогов (НДС и налог на прибыль). Дополнительно документы автоматически формируют проводки по налоговым счетам (если в настройках стоит **Способ ведения налогового учета одновременно с бухгалтерским учетом**). При вводе документа "Закрытие месяца" обратите внимание на месяц, который вы закрываете.

2.3. На рис. П3 приведен пример расчета суммы процентов по договору ДЗ-70 с помощью табло (вызывается через меню **Сервис**). На рис. П4 приведена выписка банка по погашению займа, а на рис. П5 — оборотно-сальдовая ведомость. Отрицательное сальдо (или, как говорят бухгалтеры, остаток "красным") по расчетному счету связано с тем, что мы погасили займ и проценты, не имея на то достаточных средств.

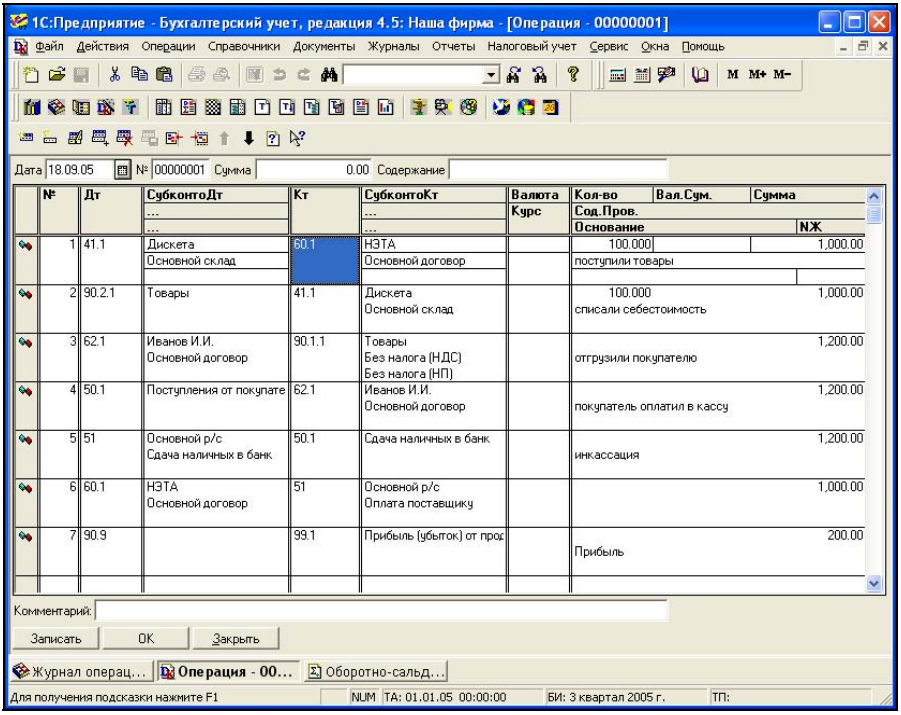


Рис. П1. Ручные проводки по рис. 2.1

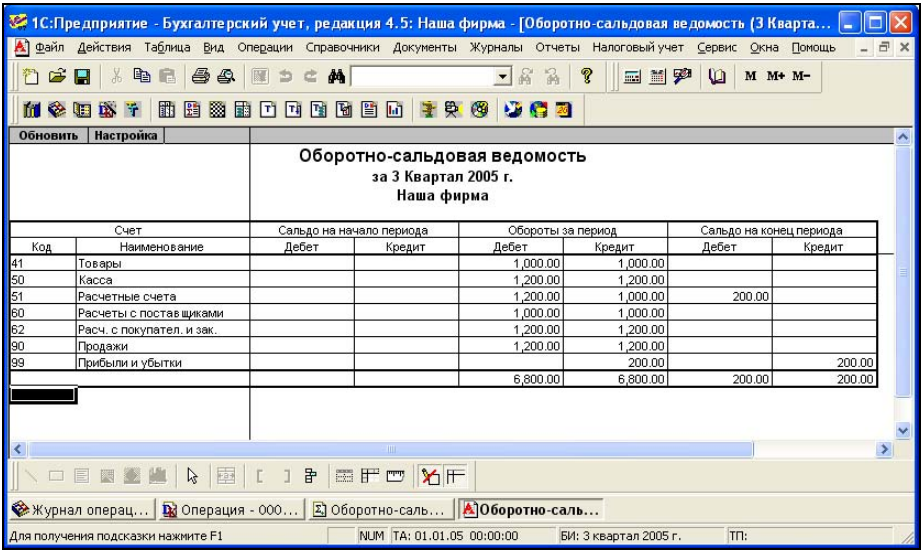


Рис. П2. Оборотно-сальдовая ведомость по рис. 2.1

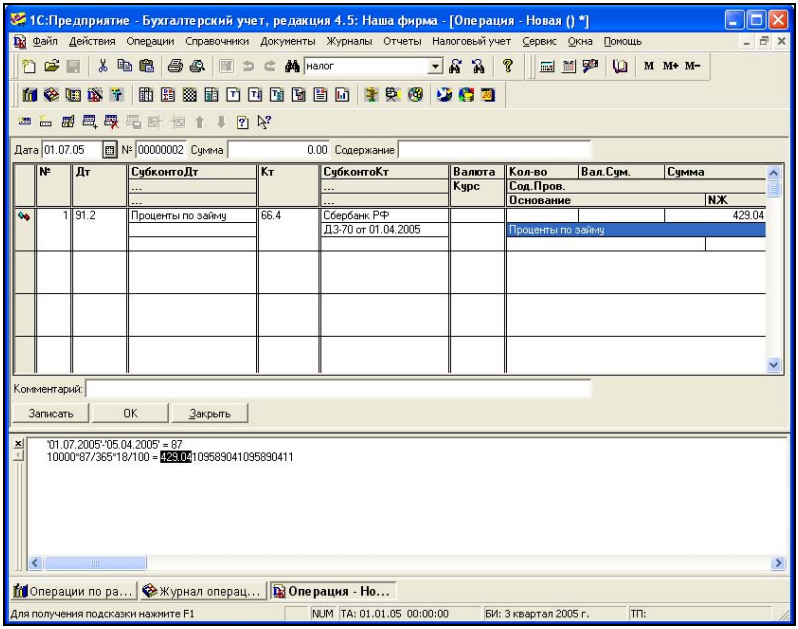


Рис. П3. Расчет суммы процентов

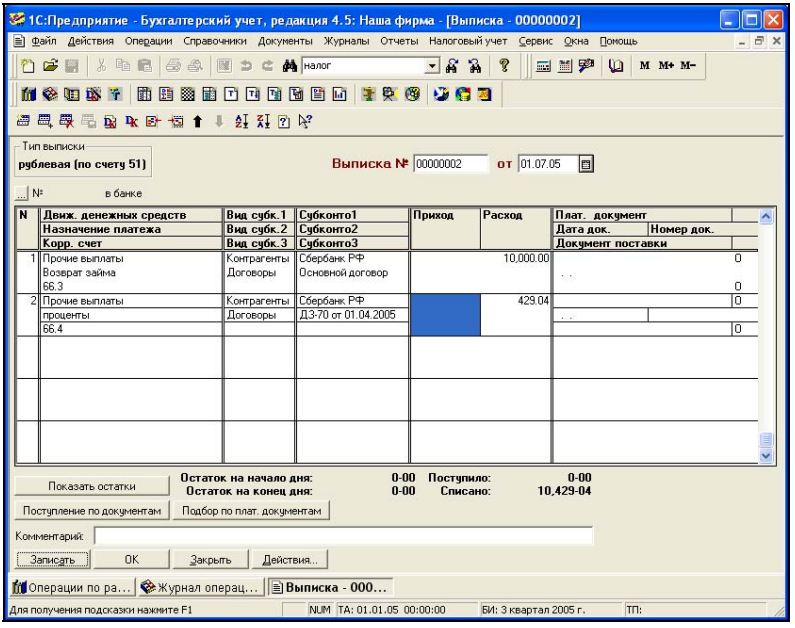


Рис. П4. Возврат займа и уплата процентов документом "Выписка"

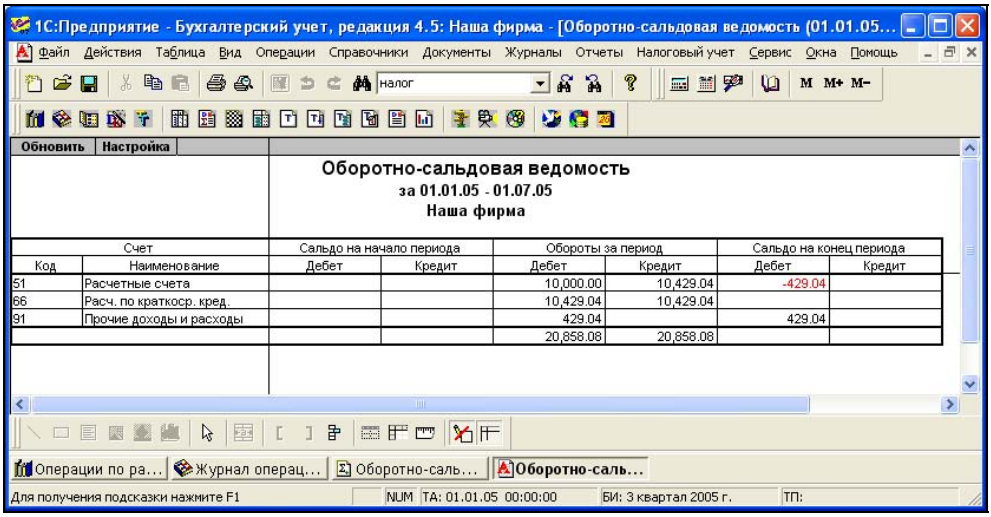


Рис. П5. Оборотно-сальдовая ведомость с отрицательным сальдо

Приведем расчет процентов по договорам займа:

- Договор ДЗ-70:

  фактический срок займа: '01.07.2005' − '05.04.2005' = 87 дней;

  сумма процентов: 10000*87/365*18/100 = 429.04109589041095890411.
- Договор ДЗ-64 в USD:

  фактический срок займа: '02.03.2005' − '03.03.2004' = 364 дня;

  сумма процентов: 1000*364/365*10/100 = 99.7260273972602739726.
- Договор ДЗ-65 в USD:

  фактический срок займа: '02.05.2005' − '01.03.2005' = 62 дня;

  сумма процентов: 1000*62/365*9/100 = 15.28767123287671232877.

При записи проводок суммы нужно округлить до двух знаков после запятой.

Важно!

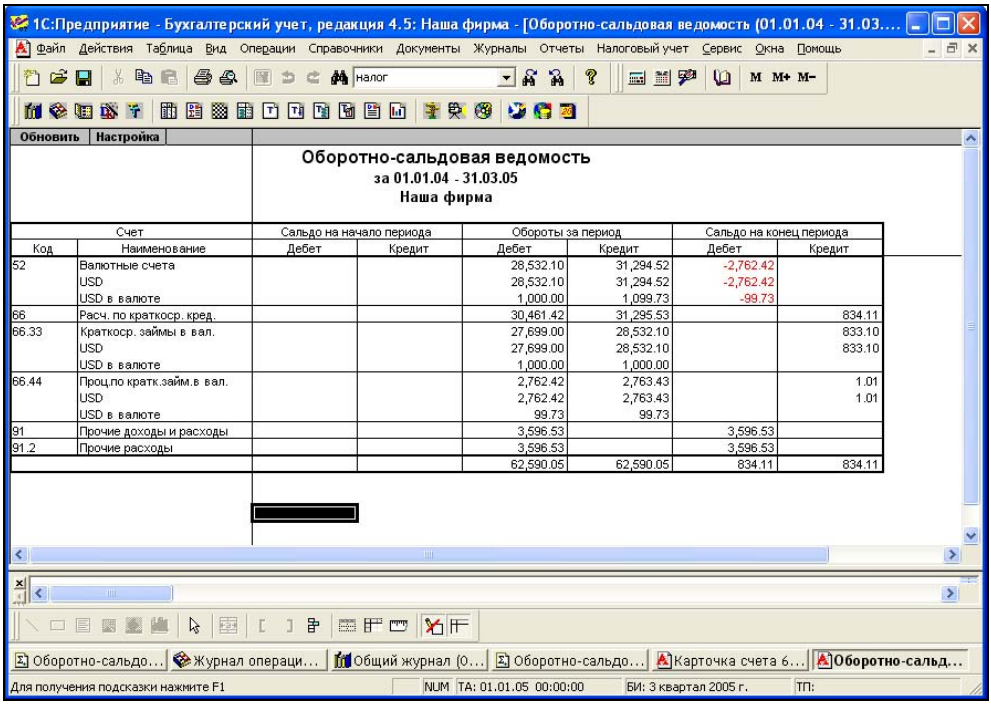
Рассмотренные примеры работы с договорами несколько упрощены. Проценты должны начисляться в конце каждого отчетного периода (Д76/К66). На затраты относятся уплаченные проценты (Д91.2/К76), то есть данной проводке должна предшествовать оплата (Д66/К51).

Для валютного договора займ должен поступить на валютный расчетный счет. После операций с валютой из-за колебания курсов валют появляется

курсовая разница: остаток по счету в валюте по текущему курсу не равен остатку по счету в рублях (рис. П6).

Совет

Курсы валют можно загрузить с сайта **www.rbc.ru** "РосБизнесКонсалтинг", запустив обработку "Загрузка курсов валют".



Счет		Сальдо на начало периода		Обороты за период		Сальдо на конец периода	
Код	Наименование	Дебет	Кредит	Дебет	Кредит	Дебет	Кредит
52	Валютные счета			28,532.10	31,294.52	-2,762.42	
	USD			28,532.10	31,294.52	-2,762.42	
	USD в валюте			1,000.00	1,099.73	-99.73	
66	Расч. по краткоср. кред.			30,461.42	31,295.53		834.11
66.33	Краткоср. займы в вал.			27,699.00	28,532.10		833.10
	USD			27,699.00	28,532.10		833.10
	USD в валюте			1,000.00	1,000.00		
66.44	Проц.по кратк.займ.в вал.			2,762.42	2,763.43		1.01
	USD			2,762.42	2,763.43		1.01
	USD в валюте			99.73	99.73		
91	Прочие доходы и расходы			3,596.53		3,596.53	
91.2	Прочие расходы			3,596.53		3,596.53	
				62,590.05	62,590.05	834.11	834.11

Рис. П6. Оборотно-сальдовая ведомость в валюте

Контрольные вопросы

1. В одной проводке балансовый и забалансовый счета корреспондировать не могут, так как тогда нарушится баланс. Система 1С:Предприятие 7.7 поддерживает это ограничение при записи операции (как вручную, так и программно).
2. Если дискеты приобретены для продажи, то это счет 41.1 (товары). Если для производства — счет 10.1 (материалы). Если организация получает их по договору комиссии, то дискеты нужно учитывать на забалансовом счете 004. Если организация производит дискеты, то они являются готовой продукцией и учитываются на счете 43.

3. Первичный документ должен иметь наименование, дату составления, наименование организации, содержание хозяйственной операции, измерители хозяйственной операции в натуральном и денежном выражении, наименование должностей ответственных лиц и их личные подписи.
4. При изменении курса валюты возникает курсовая разница — остаток по счету в валюте по текущему курсу не равен остатку по счету в рублях. Курсовая разница списывается на счет 91, как прочие доходы или расходы.
5. Применяем следующие правила (СНД — сальдо начальное дебетовое, СНК — сальдо начальное кредитовое, ДО — дебетовый оборот, КО — кредитовый оборот, СКД — сальдо конечное дебетовое, СКК — сальдо конечное кредитовое):
   - если счет активный, то СКД = СНД + ДО − КО;
   - если счет пассивный, то СКК = СНК − ДО + КО;
   - если счет активно-пассивный, то:
     - СКД = Максимум (СНД − СНК + ДО − КО,0);
     - СКК = Максимум (СНК − СНД − ДО + КО,0).
6. Сальдо активного счета не может быть кредитовым, потому что для активного счета общий дебетовый оборот превышает общий кредитовый оборот.
7. Забалансовый счет является средством учета забалансовых объектов: списанных основных средств и инвентаря; товаров, принятых на комиссию и др. Забалансовые счета можно использовать как регистры для накопления информации по какому-то объекту.

Глава 3

Задания

3.4. Для определения продолжительности работы в системе нам нужно запомнить момент входа (листинг П1). При завершении работы системы запускается предопределенная процедура, в которой мы находим разность момента выхода и момента входа в секундах и выводим в модальном окне.

Листинг П1. Определение продолжительности сеанса

```
Перем ДатаНачалаРаботы, ВремяНачалаРаботы;
// Предопределенная процедура
Процедура ПриНачалеРаботыСистемы()
Перем ЧЧ,ММ,СС;
```

```
        ДатаНачалаРаботы=ТекущаяДата();
        Предупреждение("Здравствуй,мир! Сегодня "+ДатаНачалаРаботы+" "+
        ТекущееВремя(ЧЧ,ММ,СС));
        ВремяНачалаРаботы=ЧЧ*3600+ММ*60+СС; // время в секундах
КонецПроцедуры
// Предопределенная процедура
Процедура ПриЗавершенииРаботыСистемы()
        Перем ЧЧ,ММ,СС;
        ТекущееВремя(ЧЧ,ММ,СС);
        ВремяОкончанияРаботы=ЧЧ*3600+ММ*60+СС; // время в секундах
        // определяем продолжительность в секундах
        ЧислоСекунд = (ТекущаяДата()-ДатаНачалаРаботы)*24*60*60+
                (ВремяОкончанияРаботы-ВремяНачалаРаботы);
        Предупреждение("Вы проработали "+ЧислоСекунд+" сек.");
КонецПроцедуры
```

3.7. Модуль формы отчета приведен в листинге П2. Форма диалога приведена на рис. П7.

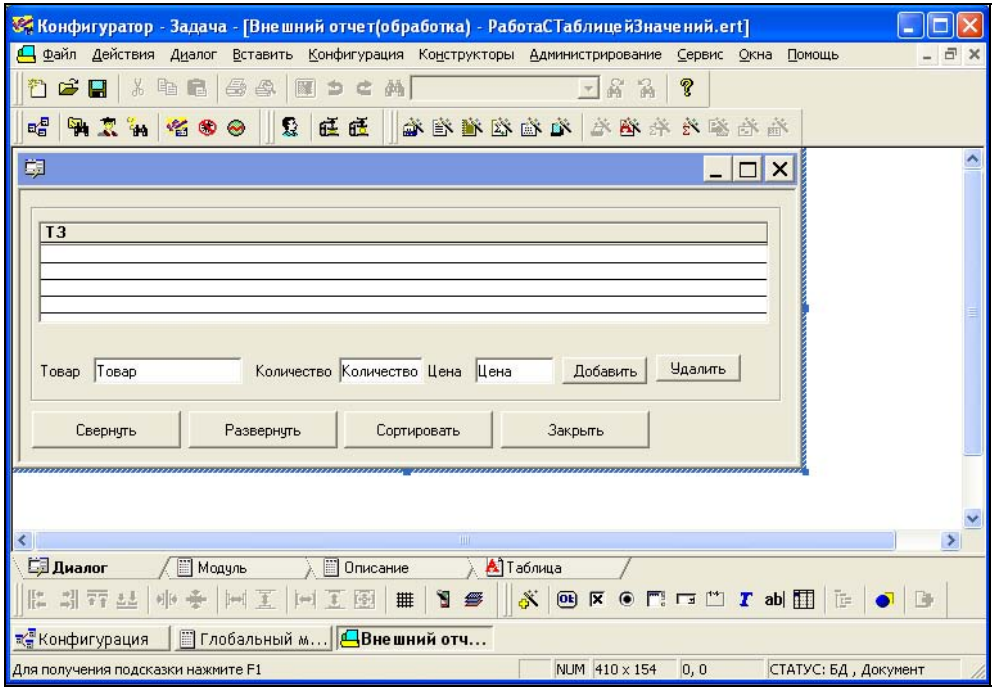


Рис. П7. Форма диалога "Работа с таблицей отчета"

Листинг П2. Модуль формы отчета "Работа с таблицей отчета"

```
Перем КопияТЗ;
// Процедура добавляет строку в таблицу
Процедура Добавить()
        ТЗ.НоваяСтрока();
        ТЗ.Товар=Товар;
        ТЗ.Количество=Количество;
        ТЗ.Цена=Цена;
        ТЗ.Сумма=Цена*Количество;
КонецПроцедуры
// Процедура удаляет строку из таблицы
Процедура Удалить()
        Если ТЗ.КоличествоСтрок()>0 Тогда
                ТЗ.УдалитьСтроку();
        КонецЕсли;
КонецПроцедуры
// Процедура сворачивает таблицу по товарам
Процедура Свернуть()
        ТЗ.Выгрузить(КопияТЗ);
        ТЗ.Свернуть("Товар","Цена,Количество,Сумма");
        // рассчитываем среднюю цену
        ТЗ.ВыбратьСтроки();
        Пока ТЗ.ПолучитьСтроку()=1 Цикл
                Если ТЗ.Количество>0 Тогда
                        ТЗ.Цена=ТЗ.Сумма/ТЗ.Количество;
                Иначе
                        ТЗ.Цена=0;
                КонецЕсли;
        КонецЦикла;
КонецПроцедуры
// Процедура восстанавливает таблицу до свертки
Процедура Развернуть()
        Если ТипЗначенияСтр(КопияТЗ)="ТаблицаЗначений" Тогда
                КопияТЗ.Выгрузить(ТЗ);
        КонецЕсли;
КонецПроцедуры
// Процедура сортирует строки таблицы по товарам и ценам
Процедура Сортировать()
```

```
        ТЗ.Сортировать("Товар+,Цена+");
КонецПроцедуры


// Создаем колонки таблицы
ТЗ.НоваяКолонка("Товар","Строка",30,,"Наименование товара",20);
ТЗ.НоваяКолонка("Цена","Число",10,2,"Цена",10);
ТЗ.НоваяКолонка("Количество","Число",10,3,"Количество",10);
ТЗ.НоваяКолонка("Сумма","Число",15,3,"Сумма",10);
```

3.10. Модуль расчета экспоненты приведен в листинге П3. Результат вычисления приведен на рис. П8. Функцию x^y можно вычислить по формуле $x^y = e^{y\,Лог(x)}$.

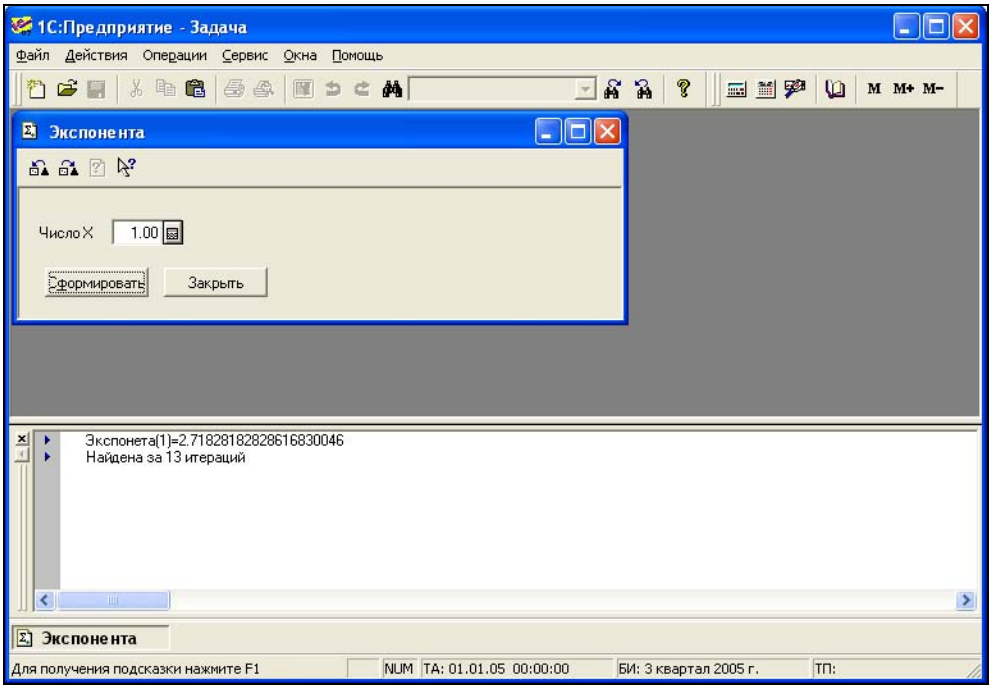


Рис. П8. Результат вычисления экспоненты от единицы

Листинг П3. Вычисление экспоненты

```
Процедура Сформировать(X)
        Н=1;
        Эксп=1;
```

```
        Приращение=1;
        Пока Приращение>0.00000001 Цикл
                Приращение=Приращение*X/H;
                Эксп=Эксп+Приращение;
                H=H+1;
        КонецЦикла;
        Сообщить("Экспонента("+X+")="+Эксп);
        Сообщить("Найдена за "+H+" итераций ");
КонецПроцедуры
```

Контрольные вопросы

1. Встроенный язык предназначен только для работы в системе 1С:Предприятие 7.7.
2. В глобальном модуле располагаются глобальные процедуры и функции, а также предопределенные процедуры и функции глобального контекста.
3. Локальный контекст совпадает по времени существования с объектом, который его образует.
4. Чтобы запустить программный модуль на выполнение, нужно открыть форму соответствующего объекта.
5. Имя переменной не может начинаться с цифры, так как в противном случае невозможно было бы интерпретировать однозначно арифметические выражения.
6. Если после слова `КонецПроцедуры` поставить точку с запятой, то система воспримет это как оператор. Если далее идет описание еще одной процедуры, то будет выдана ошибка.
7. Ключевое слово `Далее` используется для описания заголовка процедуры или функции, чтобы ее можно было вызывать из процедур или функций, идущих до основного описания.
8. Предопределенные процедуры используются для выполнения каких-либо действий при наступлении определенного события. Для правильной обработки событий важно понимать их последовательность. В случае затруднения можно воспользоваться отладчиком.
9. Если передача параметров в процедуру выполняется по ссылке, то изменение параметра внутри процедуры приведет к изменению параметра и при выходе из процедуры.
10. Если в названии процедуры или функции идет префикс `гл`, то она описана в глобальном модуле.
11. Транзакция применяется при выполнении критических действий, которые можно откатить. Так как транзакция блокирует базу данных, то выполнение операций по записи данных будет значительно быстрее.

Глава 4

Задания

4.2. Чтобы дата окончания вычислялась автоматически, нужно у реквизитов `ДатаНачала`, `СрокМесяцев` и `СрокДней` разместить на вкладке **Дополнительно** вызов формулы на встроенном языке: `ВычислитьДатуОкончания()`, текст которой приведен в листинге П4.

Листинг П4. Вычисление даты окончания

```
Процедура ВычислитьДатуОкончания()
        ДатаОкончания=ДобавитьМесяц(ДатаНачала,СрокМесяцев)+СрокДней;
КонецПроцедуры
```

4.3. Текст соответствующих процедур приведен в листингах П5 и П6.

Листинг П5. Отчет по договорам займа

```
Док=СоздатьОбъект("Документ.ДоговорЗайма");
Док.ВыбратьДокументы(НачДата,КонДата);
Пока Док.ПолучитьДокумент()=1 Цикл
        Если ВыбАгент.Выбран()=1 Тогда
                Если Док.Агент<>ВыбАгент Тогда
                        Продолжить;
                КонецЕсли;
        КонецЕсли;
        Если ВыбЗаимодавец.Выбран()=1 Тогда
                Если Док.Заимодавец<>ВыбЗаимодавец Тогда
                        Продолжить;
                КонецЕсли;
        КонецЕсли;
        Сообщить("Договор №"+Док.НомерДок);
        Сообщить("Дата начала "+Док.ДатаНачала);
        Сообщить("Дата завершения "+Док.ДатаОкончания);
        Сообщить("Срок "+Док.СрокМесяцев + " и " + Док.СрокДней);
        Сообщить("Сумма договора "+Док.Сумма + " " + Док.Валюта);
        Сообщить("Процент срочный "+Док.ПроцентСрочный);
        Сообщить("Процент досрочный "+Док.ПроцентДосрочный);
КонецЦикла;
```

Листинг П6. Изменение документов "ДоговорЗайма"

```
Док=СоздатьОбъект("Документ.ДоговорЗайма");
Док.ВыбратьДокументы(НачДата,КонДата);
Пока Док.ПолучитьДокумент()=1 Цикл
        Док.ПроцентДосрочный=Док.ПроцентСрочный/2;
        Док.Записать();
КонецЦикла;
```

Листинг П7. Создание документа "ДоговорЗайма"

```
// Создаем элемент справочника "Агенты"
Спр=СоздатьОбъект("Справочник.Агенты");
Если Спр.НайтиПоНаименованию("Иванов И.И.")=0 Тогда
        Спр.Новый();
        Спр.Наименование="Иванов И.И.";
        Спр.Записать();
КонецЕсли;
Иванов=Спр.ТекущийЭлемент();
// Создаем элемент справочника "Заимодавцы"
Спр=СоздатьОбъект("Справочник.Заимодавцы");
Если Спр.НайтиПоНаименованию("Сбербанк РФ")=0 Тогда
        Спр.Новый();
        Спр.Наименование="Сбербанк РФ";
        Спр.Записать();
КонецЕсли;
Сбербанк=Спр.ТекущийЭлемент();
Док=СоздатьОбъект("Документ.ДоговорЗайма");
Док.Новый();
Док.Сумма=1000;
Док.Агент=Иванов;
Док.Заимодавец=Сбербанк;
Док.ДатаНачала='01.01.2005';
Док.ДатаОкончания='31.10.2005';
Док.Записать();
```

Контрольные вопросы

1. Базовые объекты можно создать в любой версии "1С", а прикладные — только при наличии соответствующей компоненты.
2. Значение константы задается либо пользователем в режиме "1С:Предприятие", либо программно. Значение константы может изменяться.
3. Периодические значения могут быть заданы с точностью до даты.
4. У иерархического справочника одни элементы могут быть подчинены другим элементам (группам). Максимальная глубина вложенности иерархического справочника — 10 уровней (в стандартных версиях — 3). Родителем для всех элементов первого уровня является пустое значение.
5. Два справочника можно связать либо через механизм подчинения, либо создав реквизит типа "Справочник" соответствующего вида.
6. Ссылку на элемент справочника можно получить функцией `ТекущийЭлемент()`, ссылку на документ — функцией `ТекущийДокумент()`.
7. *См. разд. 4.4.1.*
8. Реквизиты шапки документа отличаются от реквизитов табличной части так же, как переменная отличается от массива или списка.
9. Проведение документа — формирование движений по бухгалтерским счетам, регистрам оперативного учета, журналам расчетов. В форме настройки документа нужно поставить флажок **Разрешить проведение** и указать компоненты, по которым выполняется движение.
10. *См. разд. 4.4.4.*
11. Графа отбора используется для быстрого получения документов по указанным реквизитам (обычно не более одного реквизита в документе).

Глава 5

Задания

5.2. Запрос с одновременным отбором по агентам и заимодавцам приведен в листинге П8.

Листинг П8. Запрос по договорам займа

```
Период С НачДата По КонДата;
ОбрабатыватьДокументы все;
Агент = Документ.ДоговорЗайма.Агент;
Заимодавец = Документ.ДоговорЗайма.Заимодавец;
```

```
Договор = Документ.ДоговорЗайма.ТекущийДокумент;
Группировка Договор;
Условие(Агент в ВыбАгент);
Условие(Заимодавец в ВыбЗаимодавец);
```

5.3. Формирование списка договоров с помощью конструктора запросов показано в листинге П9. Отметим, что конструктор автоматически создал также вывод результата запроса в таблицу.

Листинг П9. Запрос по договорам займа

```
Процедура ЗапросПоДоговорам()
        Перем Запрос, ТекстЗапроса, Таб;
        // Создание объекта типа "Запрос"
        Запрос = СоздатьОбъект("Запрос");
        ТекстЗапроса =
        "//{{ЗАПРОС(ЗапросПоДоговорам)
        |Период с ВыбНачПериода по ВыбКонПериода;
        |ОбрабатыватьДокументы все;
        |Обрабатывать НеПомеченныеНаУдаление;
        |Агент = Документ.ДоговорЗайма.Агент;
        |Заимодавец = Документ.ДоговорЗайма.Заимодавец;
        |Валюта = Документ.ДоговорЗайма.Валюта;
        |Договор = Документ.ДоговорЗайма.ТекущийДокумент;
        |Сумма = Документ.ДоговорЗайма.Сумма;
        |Функция СуммаДоговоров = Сумма(Сумма);
        |Группировка Валюта;
        |Группировка Договор упорядочить по Договор.НомерДок;
        |Условие(Агент в ВыбАгент);
        |Условие(Заимодавец в ВыбЗаимодавец);
        |"//}}ЗАПРОС
        ;
        // Если ошибка в запросе, то выход из процедуры
        Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
                Возврат;
        КонецЕсли;
        // Подготовка к заполнению выходных форм данными запроса
        Таб = СоздатьОбъект("Таблица");
        Таб.ИсходнаяТаблица("ЗапросПоДоговорам");
        // Заполнение полей "Заголовок"
```

```
    Таб.ВывестиСекцию("Заголовок");
    Состояние("Заполнение выходной таблицы...");
    Таб.Опции(0, 0, Таб.ВысотаТаблицы(), 0);
    Итого=0;
    Пока Запрос.Группировка(1) = 1 Цикл
        // Заполнение полей Валюта
        Таб.ВывестиСекцию("Валюта");
        Пока Запрос.Группировка(2) = 1 Цикл
            // Заполнение полей Договор
            Таб.ВывестиСекцию("Договор");
        КонецЦикла;
    КонецЦикла;
    // Заполнение полей "Итого"
    Таб.ВывестиСекцию("Итого");
    // Вывод заполненной формы
    Таб.ТолькоПросмотр(1);
    Таб.Показать("ЗапросПоДоговорам", "");
КонецПроцедуры
```

Контрольные вопросы

1. Запрос — это агрегатный тип данных, средство для извлечения и группировки данных по определенным правилам.
2. В запросе можно использовать операторы: `Период С … По`, `Группировка`, `Функция`, `Условие`, `Без итогов`, `Обрабатывать Документы`, `Обрабатывать`, а также описывать внутренние переменные.
3. Минимальный набор операторов — два: описание внутренней переменной и группировка.
4. Операторы запроса могут быть расположены в любом порядке, но внутренние переменные должны быть описаны до их использования в группировке, функции или условии.
5. В запросе можно использовать функции, описанные на встроенном языке 1С, для проверки условий, но это будет существенно замедлять выполнение запроса.
6. В арифметических выражениях запроса можно использовать функции, описанные на встроенном языке 1С, но это будет существенно замедлять выполнение запроса.

7. Оператор `Без итогов` используется для получения более быстрого результата за счет того, что не накапливаются итоги по группировкам. Отметим, что итоги по группам справочников накапливаются независимо от этого оператора.
8. Чтобы узнать количество документов или элементов справочника, надо выполнить запрос по документам или элементам справочника с использованием функции `Счётчик`, а потом посмотреть итог по всем группировкам.
9. Главное требование для быстрого выполнения запросов — не использовать в условиях и функциях выражения на встроенном языке системы 1С:Предприятие 7.7.

Глава 6

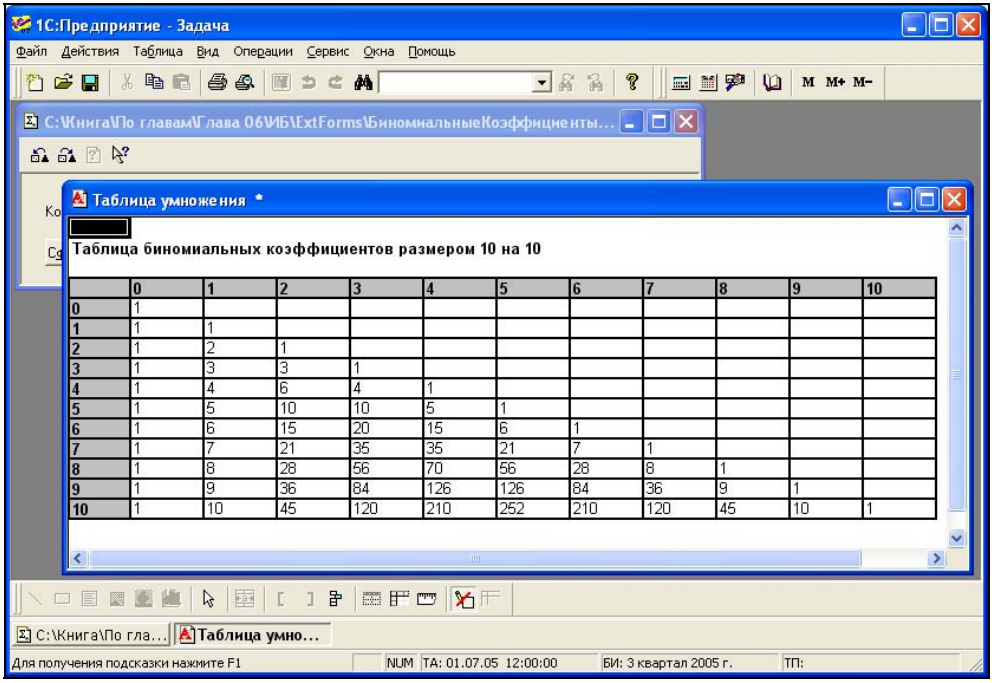
Задания

6.1. Формирование биномиальных коэффициентов приведено в листинге П10. Результат приведен на рис. П9. Процедура транспонирования приведена в листинге П11.

Листинг П10. Печать таблицы биномиальных коэффициентов

```
Функция Факториал(Знач Н)
        Если Н<=0 Тогда
                Возврат 1;
        КонецЕсли;
        Возврат Н*Факториал(Н-1);
КонецФункции
// Вычисление биномиального коэффициента
Функция БиномКоэф(Н,М)
Если М<=Н Тогда
        Возврат Факториал(Н)/Факториал(М)/Факториал(Н-М);
Иначе
        Возврат "";
КонецЕсли;
КонецФункции
// Вывод таблицы биномиальных коэффициентов
Процедура Сформировать()
Таб=СоздатьОбъект("Таблица");
```

```
Таб.ИсходнаяТаблица("БиномиальныеКоэффициенты");
// Выводим шапку
Таб.ВывестиСекцию("Шапка");
Таб.ВывестиСекцию("Заголовок|НомСтр");
Для Кол=0 По КоличествоСтолбцов Цикл
        Таб.ПрисоединитьСекцию("Заголовок|Колонка");
КонецЦикла;
// Выводим таблицу умножения
Для Стр=0 По КоличествоСтрок Цикл
        Таб.ВывестиСекцию("Строка|НомСтр");
        Для Кол=0 По КоличествоСтолбцов Цикл
                Коэффициент = БиномКоэф(Стр,Кол);
                Таб.ПрисоединитьСекцию("Строка|Колонка")
        КонецЦикла;
КонецЦикла;
Таб.ТолькоПросмотр(1);
Таб.Показать("Биномиальные коэффициенты");
КонецПроцедуры
```



	0	1	2	3	4	5	6	7	8	9	10
0	1										
1	1	1									
2	1	2	1								
3	1	3	3	1							
4	1	4	6	4	1						
5	1	5	10	10	5	1					
6	1	6	15	20	15	6	1				
7	1	7	21	35	35	21	7	1			
8	1	8	28	56	70	56	28	8	1		
9	1	9	36	84	126	126	84	36	9	1	
10	1	10	45	120	210	252	210	120	45	10	1

Рис. П9. Таблица биномиальных коэффициентов

Листинг П11. Транспонирование таблицы

```
Процедура Транспонировать(Таб)
Перем Текст, Расшифровка;
Размер=Макс(Таб.ШиринаТаблицы(),Таб.ВысотаТаблицы());
Для Кол=1 По Размер Цикл
Для Стр=Кол+1 По Размер Цикл
Текст=Таб.Область(Стр,Кол).Текст;
Расшифровка=Таб.Область(Стр,Кол).Расшифровка();
Таб.Область(Стр,Кол).Текст=Таб.Область(Кол,Стр).Текст;
Таб.Область(Стр,Кол).Расшифровка(Таб.Область(Кол,Стр).Расшифровка());
Таб.Область(Кол,Стр).Текст=Текст;
Таб.Область(Кол,Стр).Расшифровка(Расшифровка);
КонецЦикла;
КонецЦикла;
Таб.Показать();
```

6.3. Лучше всего реализовать отчет с пустой таблицей. Тогда на форме разместим две кнопки: **Сформировать** и **Записать**. Первая вызывает процедуру формирования отчета, а вторая — запись введенных пользователем процентов в документы "ДоговорЗайма". Ссылку на договор берем из расшифровки ячейки, в которой мы запоминали договор при формировании отчета. Текст программного модуля приведен в листинге П12.

Листинг П12. Запись измененных процентов в документ "ДоговорЗайма"

```
Процедура Сформировать()
// Таб=СоздатьОбъект("Таблица");
Таб=Таблица;
Таб.ИсходнаяТаблица("Отчет");
Таб.Очистить();
Таб.ВывестиСекцию("Шапка");
Дог=СоздатьОбъект("Документ.ДоговорЗайма");
Дог.ВыбратьДокументы(НачДата,КонДата);
Пока Дог.ПолучитьДокумент()=1 Цикл
       Таб.ВывестиСекцию("Договор");
КонецЦикла;
Таб.Показать();
КонецПроцедуры
// Запись новых процентов
```

```
Процедура ЗаписатьИзменения()
НомерСтроки=4;
Пока 1=1 Цикл
        Обл=Таблица.Область(НомерСтроки,7);
        ПроцентСрочный=Число(Обл.Текст);
        Обл=Таблица.Область(НомерСтроки,8);
        ПроцентДосрочный=Число(Обл.Текст);
        ТекДог=Обл.Расшифровка();
        Если ПустоеЗначение(ТекДог)=1 Тогда
                Прервать;
        КонецЕсли;
        Дог=СоздатьОбъект("Документ.ДоговорЗайма");
        Если Дог.НайтиДокумент(ТекДог)=1 Тогда
                Дог.ПроцентСрочный=ПроцентСрочный;
                Дог.ПроцентДосрочный=ПроцентДосрочный;
                Дог.Записать();
        КонецЕсли;
        НомерСтроки=НомерСтроки+1;
КонецЦикла;
Таблица.Показать();
КонецПроцедуры
```

Контрольные вопросы

1. Таблица в системе 1С:Предприятие 7.7 используется для формирования отчетов и вывода их на печать, для хранения таблиц-шаблонов и для организации ввода в регламентированные формы с большим количеством данных.
2. Существует два режима работы с таблицей: "обычный" и "для ввода данных".
3. Для чтения-записи значений из ячейки используется функция таблицы `Область()`. В обычном режиме у области есть реквизит `Текст`, а в режиме ввода данных — реквизит `Значение`. Кроме того, в режиме ввода данных можно обращаться к ячейкам по именам и загружать/выгружать значения незащищенных ячеек с помощью списка значений.
4. Таблицы-шаблоны применяются для визуального форматирования таблиц. Когда таблица достаточно сложная, то это дает выигрыш во времени создания отчета. Недостаток форматного вывода в том, что вывод идет в потоковом режиме: сверху вниз и слева направо, то есть вернуться назад и вставить секцию уже невозможно. Поэтому на практике часто комбинируют эти два способа заполнения таблицы.

5. Механизм обновления таблиц работает через использование предопределенной процедуры `ОбработкаЯчейкиТаблицы()`. Отчет или обработка должны быть встроенными в конфигурацию.
6. Чтобы совместить в одной форме реквизиты диалога и таблицу, нужно в свойствах формы указать: "использовать таблицу" — "пустую" или "для ввода данных".
7. В режиме ввода данных таблицы могут динамически обновляться, как в MS EXCEL, что позволяет их использовать в качестве форм, где часть данных нужно вводить, а часть — должна вычисляться.

Глава 7

Задания

7.1. Для того чтобы счету можно было задать признак "Валютный", необходимо, чтобы в настройках плана счетов был выбран справочник валют. Там же необходимо задать максимальное количество субконто — три. При добавлении субсчета, например, 91.2, счет-родитель 91 создается автоматически.

7.2. Текст процедуры, формирующей отчет, приведен в листинге П13, а таблица-шаблон — на рис. П10.

Листинг П13. Отчет по проводкам

```
Процедура Сформировать()
Таб=СоздатьОбъект("Таблица");
Таб.ВывестиСекцию("Шапка");
Таб.ВывестиСекцию("Заголовок|Дата");
Тип="Дебет";Таб.ПрисоединитьСекцию("Заголовок|Счет");
Тип="Кредит";Таб.ПрисоединитьСекцию("Заголовок|Счет");
Таб.ПрисоединитьСекцию("Заголовок|Суммы");
Оп=СоздатьОбъект("Операция");
Оп.ВыбратьОперацииСПроводками(НачДата,КонДата,Фильтр);
Пока Оп.ПолучитьПроводку()=1 Цикл
        ПечДата=Оп.ДатаОперации;
        Таб.ВывестиСекцию("Проводка|Дата");
        ПечСчет=Оп.Дебет.Счет;
        ПечСубк1=Оп.Дебет.Субконто(1);
        ПечСубк2=Оп.Дебет.Субконто(2);
        ПечСубк3=Оп.Дебет.Субконто(3);
```

```
        Таб.ПрисоединитьСекцию("Проводка|Счет");
        ПечСчет=Оп.Кредит.Счет;
        ПечСубк1=Оп.Кредит.Субконто(1);
        ПечСубк2=Оп.Кредит.Субконто(2);
        ПечСубк3=Оп.Кредит.Субконто(3);
        Таб.ПрисоединитьСекцию("Проводка|Счет");
        ПечВалюта=Оп.Валюта;
        ПечВалСумма=Оп.ВалСумма;
        ПечСумма=Оп.Сумма;
        ПечКоличество=Оп.Количество;
        Таб.ПрисоединитьСекцию("Проводка|Суммы");
КонецЦикла;
Таб.Показать();
КонецПроцедуры
```

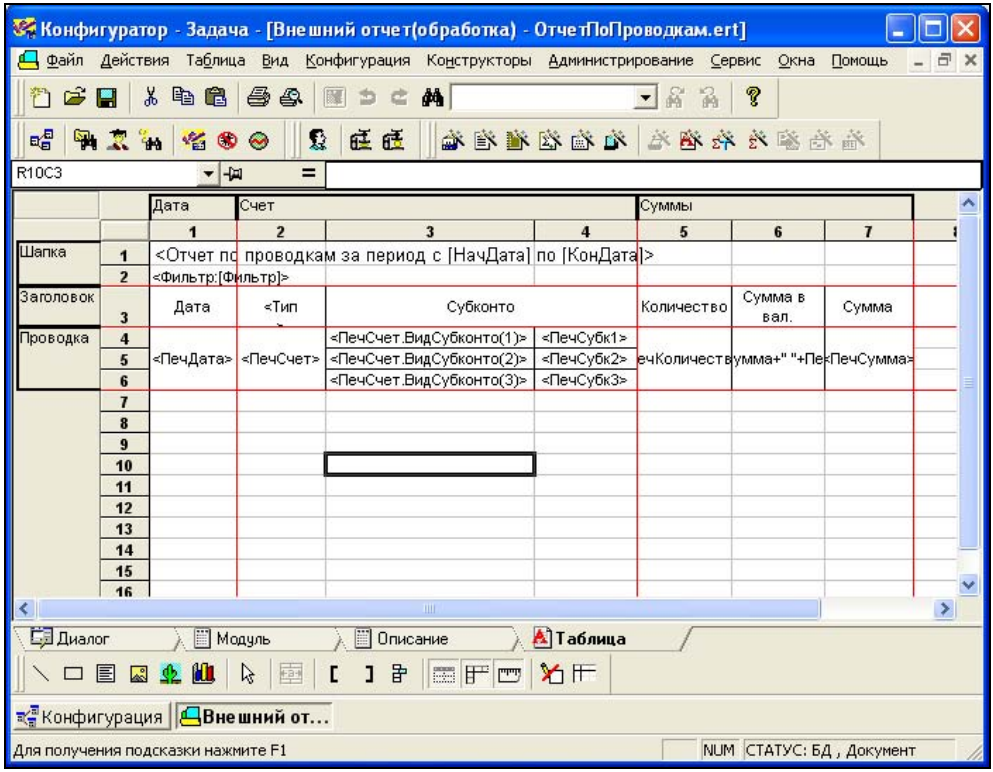


Рис. П10. Таблица-шаблон отчета по проводкам

7.3. Модуль документа "ЗакрытиеДоговора" приведен в листинге П14, а отчет по договорам, завершенным досрочно, — в листинге П15.

Листинг П14. Модуль документа "ЗакрытиеДоговора"

```
Процедура ОбработкаПроведения()
// Определяем сумму займа
Оп=СоздатьОбъект("Операция");
// Задаем фильтр по субконто
Оп.ИспользоватьСубконто(ВидыСубконто.ДоговорыЗайма,Договор);
Оп.ВыбратьОперацииСПроводками(Договор.ДатаНачала,ДатаДок,"51,66.3;52,66.3
3");
Сумма=0;
Пока Оп.ПолучитьПроводку()=1 Цикл
        Если (Оп.Дебет.Счет=СчетПоКоду("52")) и (Оп.ВалСумма>0) Тогда
                // Сумма в валюте
                Сумма=Оп.ВалСумма;
                ДатаНачала=Оп.ДатаОперации;
                Валюта=Оп.Валюта;
                Агент=Оп.Кредит.Субконто(1);
                Заимодавец=Оп.Кредит.Субконто(2);
                Прервать;
        ИначеЕсли (Оп.Дебет.Счет=СчетПоКоду("51")) и (Оп.Сумма>0) Тогда
                // Сумма в рублях
                Сумма=Оп.Сумма;
                ДатаНачала=Оп.ДатаОперации;
                Валюта=Константа.БазоваяВалюта;
                Агент=Оп.Кредит.Субконто(1);
                Заимодавец=Оп.Кредит.Субконто(2);
                Прервать;
        КонецЕсли;
КонецЦикла;
Если Сумма=0 Тогда
        Сообщить("По договору "+Договор+" не поступали денежные
средства!");
        НеПроводитьДокумент();
        Возврат;
КонецЕсли;
// Определяем срок договора, процент
```

```
Срок=ДатаДок-ДатаНачала;
Если ДатаДок<Договор.ДатаОкончания Тогда
        Процент=Договор.ПроцентДосрочный;
Иначе
        Процент=Договор.ПроцентСрочный;
КонецЕсли;
// Определяем сумму процентов в валюте договора
СуммаПроцентов=Сумма*Процент*Срок/365/100;
// Создаем проводку
Операция.НоваяПроводка();
Операция.Дебет.Счет=СчетПоКоду("91.2");
Если Валюта=Константа.БазоваяВалюта Тогда
        // Договор в рублях
        Операция.Кредит.Счет = СчетПоКоду("66.4");
        Операция.Сумма=СуммаПроцентов;
Иначе
        // Договор в валюте
        Операция.Кредит.Счет = СчетПоКоду("66.44");
        Операция.ВалСумма=СуммаПроцентов;
        Операция.Валюта=Валюта;
        Операция.Сумма=СуммаПроцентов*
                Валюта.Курс.Получить(ДатаДок)/
                Валюта.Кратность.Получить(ДатаДок);
КонецЕсли;
Операция.Кредит.Агенты=Агент;
Операция.Кредит.Заимодавцы=Заимодавец;
Операция.Кредит.ДоговорыЗайма=Договор;
Операция.Записать();
КонецПроцедуры
```

Листинг П15. Отчет по договорам займа, завершенным досрочно

```
Процедура Сформировать()
Таб=СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("Отчет");
Таб.ВывестиСекцию("Шапка");
Оп=СоздатьОбъект("Операция");
Оп.ВыбратьОперацииСПроводками(НачДата,КонДата,"66.4,66.44,91.2");
Пока Оп.ПолучитьПроводку()=1 Цикл
```

```
        Если (Оп.Дебет.Счет=СчетПоКоду("91.2"))
                и ((Оп.Кредит.Счет=СчетПоКоду("66.4"))
                или (Оп.Кредит.Счет=СчетПоКоду("66.44")))  Тогда
                Договор=Оп.Кредит.Субконто(3);
                Если Договор.ДатаОкончания>Оп.ДатаОперации Тогда
                        Таб.ВывестиСекцию("Договор");
                КонецЕсли;
        КонецЕсли;
КонецЦикла;
Таб.Показать();
КонецПроцедуры
```

Контрольные вопросы

1. Иерархия счетов образуется по коду счета. Например, для счета с кодом 41.1 родителем является счет 41. Ярлычки групп счетов при просмотре плана счетов — желтые, а ярлычки счетов, которые могут использоваться в проводках, — синие. Кроме того, если на ярлычке есть красная галочка, то этот счет введен в конфигураторе.
2. У "ручной" операции вид документа "Операция".
3. Порядок заполнения реквизитов проводки важен при определении субконто. Нельзя задавать субконто, не задав счет.
4. Номер операции хранится в связанном с ней документе.
5. Чтобы перебрать все проводки вида Д41.1—К60.1, нужно воспользоваться функцией `ВыбратьОперацииСПроводками()`.
6. У субконто (при привязке его к счету) есть признаки:
   - **Только обороты** — если установить этот признак, то по этому субконто не будут разворачиваться остатки счета, а будут разворачиваться только обороты;
   - **Учет по сумме**, **Учет по валютной сумме**, **Учет по количеству** — эти признаки позволяют регулировать использование данного субконто с точки зрения суммового, валютного и количественного учета.

Глава 8

Задания

8.1. Отчет о задержке в поступлении денежных средств приведен в листинге П16.

Листинг П16. Отчет о задержке в поступлении денежных средств

```
Таб=СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("Отчет");
Таб.ВывестиСекцию("Шапка");
Дог=СоздатьОбъект("Документ.ДоговорЗайма");
Дог.ВыбратьДокументы(НачДата,КонДата);
Пока Дог.ПолучитьДокумент()=1 Цикл
        ТекДог=Дог.ТекущийДокумент();
        БИ=СоздатьОбъект("БухгалтерскиеИтоги");
        БИ.ИспользоватьСубконто("ДоговорыЗайма",ТекДог);
        БИ.ВыполнитьЗапрос(НачДата,ТекущаяДата(),
                "66.3,66.33","51,52",,,"Проводка");
        БИ.ВыбратьПериоды();
        ДатаПоступления="Отсутствует";
        Пока БИ.ПолучитьПериод()=1 Цикл
                Если (БИ.Операция.Дебет.Счет=СчетПоКоду("51"))
                        или (БИ.Операция.Дебет.Счет=СчетПоКоду("52"))
                        Тогда
                        ДатаПоступления=БИ.Операция.ДатаОперации;
                        Прервать;
                КонецЕсли;
        КонецЦикла;
        Если (ТипЗначенияСтр(ДатаПоступления)="Дата") Тогда
                Если ДатаПоступления<=ТекДог.ДатаНачала Тогда
                        Продолжить;
                КонецЕсли;
        КонецЕсли;
        Таб.ВывестиСекцию("Договор");
КонецЦикла;
Таб.Показать();
```

Контрольные вопросы

1. Периодичность бухгалтерских итогов — один месяц.
2. В бухгалтерских итогах хранятся остатки и обороты по счетам и субконто, обороты между счетами.
3. При работе с бухгалтерскими итогами есть три режима: основных итогов, временного расчета и запроса.

4. Развернутое сальдо имеет смысл для активно-пассивных счетов. Например, при расчетах с покупателями свернутое сальдо показывает общий долг покупателей, если сальдо дебетовое; или наш долг покупателям, если сальдо кредитовое. На самом деле, по каждому контрагенту есть свой долг, и имеет смысл говорить, что столько-то должны мы, и столько-то должны нам. Развернутое сальдо как раз и показывает суммы и дебетового, и кредитового сальдо. При работе с бухгалтерскими итогами есть функции `СНДР()`, `СКДР()` и т. д., дающие развернутое сальдо по субсчетам; а также функции `СНДРС()`, `СКДРС()` и т. д., дающие развернутое сальдо по субконто.
5. Сальдо по оборотному субконто получить нельзя.
6. *См. разд. 8.4.3.*
7. Чтобы выполнить запрос на позицию документа, нужно указать в качестве одной из дат позицию документа, которую можно получить, например, функцией `ТекущийДокумент()`.

Глава 9

Задания

9.1. Задача решается аналогично задаче об инвентаризации материалов: нужно заменить функцию `БИ.ИспользоватьСубконто("Материалы")` на `БИ.ИспользоватьСубконто("Номенклатура")` и изменить фильтр по счетам с "10" на "41.1".

9.2. Для примера сделать вариант формирования отчета с детализацией по валютам. В свойствах формы указываем "Использовать таблицу пустую". На форме диалога помещаем две кнопки: **Кратко** (вызывается функция `Сформировать("Кратко")`) и **Детально** (вызывается функция `Сформировать("Подробно")`). Текст процедуры приведен в листинге П17.

Листинг П17. Отчет по высоколиквидным активам

```
Процедура Сформировать(Режим)
КурсОценкиНач=ВалютаОценки.Курс.Получить(НачДата)/
        ВалютаОценки.Кратность.Получить(НачДата);
КурсОценкиКон=ВалютаОценки.Курс.Получить(КонДата)/
        ВалютаОценки.Кратность.Получить(КонДата);
Таблица.Очистить();
Таблица.ВывестиСекцию("Шапка");
БИ=СоздатьОбъект("БухгалтерскиеИтоги");
```

```
БИ.ВыполнитьЗапрос(НачДата, КонДата,
                "50.1,50.11,51,52, 58.2,58.3,58.11,58.22,58.33");
БИ.ВыбратьСчета();
Пока БИ.ПолучитьСчет()=1  Цикл
    Счет=БИ.Счет;
       Если Счет.Валютный=1 Тогда
              СумР1=0; СумР2=0;
           БИ.ВыбратьВалюты();
              Пока БИ.ПолучитьВалюту()=1 Цикл
                  Валюта=БИ.Валюта;
                     В1=БИ.СНД("В");
                     Р1=В1*Валюта.Курс.Получить(НачДата)/
                            Валюта.Кратность.Получить(НачДата);
                     В2=БИ.СКД("В");
                     Р2=В2*Валюта.Курс.Получить(КонДата)/
                            Валюта.Кратность.Получить(КонДата);
                     Если Режим="Детально" Тогда
                            В1=Р1/КурсОценкиНач;
                            В2=Р2/КурсОценкиКон;
                            Таблица.ВывестиСекцию("Валюта");
                     КонецЕсли;
                     СумР1=СумР1+Р1;
                     СумР2=СумР2+Р2;
              КонецЦикла;
       Иначе
              СумР1=БИ.СНД("С");
              СумР2=БИ.СКД("С");
       КонецЕсли;
       СумВ1=СумР1/КурсОценкиНач;
       СумВ2=СумР2/КурсОценкиКон;
       Таблица.ВывестиСекцию("Строка");
КонецЦикла;
Таблица.Показать();
КонецПроцедуры
```

9.3. Портфель ценных бумаг и отчет о приобретении ценных бумаг реализуются очень просто по бухгалтерским итогам на счете 58. Отчет о реализации ценных бумаг более проблематичен. В самом деле, аналитический разрез по субконто "ЦенныеБумаги" есть на 58-м счете, но там учет ведется по себестоимости. На 76-м счете мы видим сумму выручки от реализации, но

там нет аналитического разреза по ценным бумагам. Как вариант решения — создать забалансовый счет "РЦБ", на котором отражать реализацию ценных бумаг по продажной стоимости в разрезе агентов и ценных бумаг. Тогда в модуль документа добавятся строки, показанные в листинге П18, а отчет показан в листинге П19.

Листинг П18. Учет реализации ценных бумаг по продажной стоимости на забалансовом счете

```
Операция.НоваяПроводка();
Операция.Кредит.Счет=СчРЦБ;
Операция.Кредит.Субконто(1,ЦеннаяБумага);
Операция.Кредит.Субконто(2,Агент);
Операция.Количество=Количество;
Операция.Сумма=Сумма;
```

Листинг П19. Отчет о реализации ценных бумаг за период

```
Процедура Сформировать()
Таб=СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("Таблица");
Ит=СоздатьОбъект("БухгалтерскиеИтоги");
Ит.ИспользоватьСубконто(ВидыСубконто.ЦенныеБумаги);
Ит.ИспользоватьСубконто(ВидыСубконто.Агенты);
Ит.ВыполнитьЗапрос(НачДата,КонДата,"РЦБ");
Таб.ВывестиСекцию("Заголовок");
Ит.ВыбратьСубконто(1);
Пока Ит.ПолучитьСубконто(1)=1 Цикл
        Ит.ВыбратьСубконто(2);
        Пока Ит.ПолучитьСубконто(2)=1 Цикл
                Таб.ВывестиСекцию("Строка");
        КонецЦикла;
КонецЦикла;
Таб.Показать("Сформировать","");
КонецПроцедуры
```

Контрольные вопросы

1. Инвентаризация — это сверка учетных и фактических остатков товаров материалов, денежных средств и т. п. Инвентаризация не может проводиться одновременно с работой склада, так как остатки на момент ин-

вентаризации должны быть фиксированы (в магазинах обычно вывешивают табличку "Учет").

2. Для этого нужно перебрать все элементы справочника "Материалы".
3. Нельзя оценивать высоколиквидные активы в иностранной валюте с помощью функций `СумР=БИ.СНД("С")` и `СумР=БИ.СКД("С")`, потому что из-за курсовой разницы эти суммы могут не соответствовать валютным по курсу на дату оценки.
4. У субконто "Агенты" на 58-м счете должен быть установлен признак `Только обороты`, так как приход ценных бумаг может быть от одного агента, а расход сделан другим агентом, и в этом случае остаток по агентам теряет смысл.

Глава 10

Задания

10.2. Формирование движений по регистру "Продажи" приведено в листинге П20.

Листинг П20. Проведение по регистру "Продажи"

```
ВыбратьСтроки();
Пока ПолучитьСтроку() = 1 Цикл
        // Движение по регистру Продажи
        Регистр.Продажи.Товар = Товар;
        Регистр.Продажи.Контрагент = Контрагент;
        Регистр.Продажи.Количество = Количество;
        Регистр.Продажи.Сумма = Сумма;
        Регистр.Продажи.ДвижениеВыполнить();
КонецЦикла;
```

10.4. Документ "ПеремещениеТоваров" будет иметь два реквизита шапки: `СкладИсточник` и `СкладПриемник`, и два реквизита табличной части: `Товар` и `Количество`. Модуль документа "ПеремещениеТоваров" приведен в листинге П21.

Листинг П21. Модуль документа "ПеремещениеТоваров"

```
Процедура ОбработкаПроведения()
Рег = СоздатьОбъект("Регистр.ОстаткиТоваров");
Если ИтогиАктуальны()=0 Тогда
```

```
        // Делаем временный расчет
        СписокТоваров=СоздатьОбъект("СписокЗначений");
        ВыгрузитьТабличнуюЧасть(СписокТоваров,"Товар");
        Рег.ВременныйРасчет();
        // Устанавливаем фильтр по вхождению значений измерения
        // Товар в список значений
        Рег.УстановитьЗначениеФильтра("Товар",СписокТоваров,2);
        РассчитатьРегистрыНа(ТекущийДокумент());
КонецЕсли;
ВыбратьСтроки();
Пока ПолучитьСтроку() = 1 Цикл
        // Рассчитываем остатки по текущему товару
        Рег.Остатки(Товар,СкладИсточник);
        ОстКол = Рег.Количество;
        ОстСум = Рег.Сумма;
        Если ОстКол>Количество Тогда
                // Рассчитываем среднюю себестоимость
                Себестоимость=ОстСум/ОстКол*Количество;
        ИначеЕсли ОстКол=Количество Тогда
                // Если списываем последнюю единицу товара,
                // то, чтобы избавиться от ошибок округления,
                // за себестоимость принимаем всю оставшуюся сумму
                Себестоимость=ОстСум;
        Иначе
                Сообщить("В наличии "+ОстКол+" ед. товара "+Товар+" на
                складе "+СкладИсточник+" из требуемых "+Количество);
                НеПроводитьДокумент();
                Возврат;
        КонецЕсли;
        // Списываем товар со склада-источника
        Регистр.ОстаткиТоваров.Товар = Товар;
        Регистр.ОстаткиТоваров.Склад = СкладИсточник;
        Регистр.ОстаткиТоваров.Количество = Количество;
        Регистр.ОстаткиТоваров.Сумма = Себестоимость;
        Регистр.ОстаткиТоваров.ДвижениеРасходВыполнить();
        // Оприходуем товар на склад-приемник
        Регистр.ОстаткиТоваров.Товар = Товар;
        Регистр.ОстаткиТоваров.Склад = СкладПриемник;
        Регистр.ОстаткиТоваров.Количество = Количество;
```

```
        Регистр.ОстаткиТоваров.Сумма = Себестоимость;
        Регистр.ОстаткиТоваров.ДвижениеПриходВыполнить();
КонецЦикла;
КонецПроцедуры
```

10.5. Отчет по продажам приведен в листинге П22.

Листинг П22. Отчет по продажам товаров

```
Процедура Сформировать()
// Создание объекта типа "Запрос
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса= "";
Если ВыбКонПериода>=ПолучитьДатуТА() Тогда
        ТекстЗапроса = ТекстЗапроса + "Период С ВыбНачПериода;";
Иначе
        ТекстЗапроса = ТекстЗапроса +
        "Период С ВыбНачПериода По ВыбКонПериода;";
КонецЕсли;
ТекстЗапроса = ТекстЗапроса+
        "//{{ЗАПРОС(ПоПродажам)
        |Контрагент = Регистр.Продажи.Контрагент;
        |Товар = Регистр.Продажи.Товар;
        |Количество = Регистр.Продажи.Количество;
        |Сумма = Регистр.Продажи.Сумма;
        |ДокументДвижения = Регистр.Продажи.ТекущийДокумент;
        |Функция ПродажиСумма = Сумма(Количество);
        |Функция ПродажиКоличество = Сумма(Количество);
        |Группировка Контрагент;
        |Группировка Товар;
        |Группировка ДокументДвижения;
        |Условие(Контрагент в ВыбКонтрагент);
        |Условие(Товар в ВыбТовар);
        |"//}}ЗАПРОС
        ;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
        Возврат;
КонецЕсли;
// Подготовка к заполнению выходных форм данными запроса
```

```
Таб = СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("ПоПродажам");
// Заполнение полей "Заголовок"
Таб.ВывестиСекцию("Заголовок");
Пока Запрос.Группировка(1) = 1 Цикл
        // Заполнение полей "Контрагент"
        Таб.ВывестиСекцию("Контрагент");
        Пока Запрос.Группировка(2) = 1 Цикл
                // Заполнение полей "Товар"
                Таб.ВывестиСекцию("Товар");
                Пока Запрос.Группировка(3) = 1 Цикл
                        // Заполнение полей ДокументДвижения
                        Таб.ВывестиСекцию("ДокументДвижения");
                КонецЦикла;
        КонецЦикла;
КонецЦикла;
// Заполнение полей "Итого"
Таб.ВывестиСекцию("Итого");
// Вывод заполненной формы
Таб.ТолькоПросмотр(1);
Таб.Показать("ПоПродажам", "");
КонецПроцедуры
```

10.6. Документы будут проведены, однако в результате по документу от 15.01 будет неверно определена себестоимость товара. Точка актуальности останется на 15.01. Дополнительный документ от 08.01 тоже будет проведен, но ситуация будет еще хуже: количественный итог по регистру будет неверным.

10.7. Документ "ПоступлениеТоваров" не надо перепроводить при восстановлении последовательности, потому что он не использует информацию из регистров.

10.8. Сначала создаем последовательность `ЦенныеБумаги`:

- движения, влияющие на последовательность, — счет 58.10;
- документы, которые нужно перепровести, — "ПриходЦБ" и "РасходЦБ".

В модуле формы документов "ПриходЦБ" и "РасходЦБ" нужно описать предопределенную процедуру `ПриЗаписи()`, которая приведена в листинге П23.

Листинг П23. Предопределенная процедура `ПриЗаписи ()`

```
Процедура ПриЗаписи()
// Записываем объект, чтобы нарушилась последовательность
Записать();
// Определяем следующий документ последовательности
Послед=Последовательность.ЦенныеБумаги;
НачДок=СформироватьПозициюДокумента(Послед.ПолучитьДокумент(),1)
Док=СоздатьОбъект("Документ");
Док.ВыбратьДокументы(НачДок);
Пока Док.ПолучитьДокумент()=1 Цикл
        Если Док.ПринадлежитПоследовательности("ЦБ")=1 Тогда
                Док.СделатьНеПроведенным();
        КонецЕсли;
КонецЦикла;
КонецПроцедуры
```

Контрольные вопросы

1. Компонента "Оперативный учет" предназначена для решения учетных задач в оперативном режиме. Механизмы этой компоненты наиболее эффективны при оперативном вводе информации.
2. Для конфигурирования в компоненте "Оперативный учет" специфическим объектом является "Регистр".
3. Регистры предназначены для накопления сводной информации. Итоги по регистру хранятся с заданной периодичностью в специальном файле.
4. Периодичность хранения остатков задается пользователем, и чем длиннее этот период, тем медленнее будут рассчитываться итоги на промежуточные даты.
5. Движения в регистры записываются только при проведении документов, принадлежащих компоненте "Оперативный учет".
6. Регистры остатков в отличие от оборотных регистров накапливают остатки на определенные моменты времени. Зная остатки на начало и на конец периода, можно легко вычислить оборот за период (приход и расход). Регистр оборотов накапливает только обороты.
7. По аналогии с запросами, измерения — это группировки сводной информации, ресурсы — функции, а реквизиты регистра — внутренние переменные.

8. Выборку по активным (ненулевым) остаткам регистра можно получить с помощью функций `ВыбратьИтоги()` и `ПолучитьИтог()`.
9. Выборку по движениям регистров можно получить с помощью функций `ВыбратьДвижения()` и `ПолучитьДвижение()`. Для уменьшения размера выборки применяется функция `УстановитьЗначениеФильтра()`.
10. Для оптимизации выборки итогов и движений регистров можно использовать графы отбора.
11. Функции и процедуры получения итогов по оборотным регистрам отличаются от функций и процедур получения остатков по регистрам остатков тем, что остатки выдаются на точку актуальности, а итог можно получить за указанный период, совпадающий с периодичностью оборотного регистра (или больший).
12. Временный расчет выполняется для получения итогов на произвольный момент времени. Временный расчет необходимо использовать при проведении документов "задним числом".
13. Формирование движений регистров производится функциями `ДвижениеВыполнить()` (для оборотного регистра), `ДвижениеПриходВыполнить()`, `ДвижениеРасходВыполнить()` — (для регистра остатков) .
14. Структура выборки в запросе по остаткам регистра определяется группировками и функциями запроса. В результате запроса будут получены только ненулевые итоги, полученные функциями `НачОст()`, `Приход()`, `Расход()`, `КонОст()`. В выражении "Запрос.xxxxxxx" после успешного выполнения такого запроса можно указывать названия группировок, функций и внутренних переменных запроса.
15. Наличие секции "функция" в тексте запроса является обязательным для получения непустой выборки запросом по регистру оперативного учета.

Глава 11

Контрольные вопросы

1. Компонента "Расчет" предназначена для выполнения сложных периодических расчетов. Кроме расчета заработной платы это может быть расчет квартплаты, начисление дивидендов, расчет амортизации основных средств. Специфическими объектами для "Расчета" являются виды и группы расчетов, календари, журналы расчетов.
2. Календари предназначены для определения календаря рабочего времени. При выборе стартовой даты календаря нужно учитывать периодичность календаря.

3. Пользователи могут вводить продолжительность рабочего времени по датам календаря. Автоматическое заполнение календаря выполняется на основании информации о календаре, введенной в конфигураторе. Календари могут редактироваться при помощи встроенного языка.
4. Объект "Праздники" нужен для задания исключений из правил. Объект "Праздники" в системе единственный.
5. Вид расчета предназначен для описания специфического алгоритма расчета. Виды расчета упорядочиваются в журнале расчетов по приоритету (реквизит "Очередность").
6. Группы расчетов предназначены для объединения видов расчета по какому-то признаку. С помощью групп расчета можно формировать базу вида расчета.
7. Настройка вытеснения одного вида расчета другим гарантирует, что они не будут действовать одновременно. Механизм автоматического сторнирования записей журнала расчетов выполняется при вводе вытесняющих записей с прошлым периодом действия.
8. Журналы расчетов служат для накопления актов расчета (записей расчета). Журналов расчетов может быть несколько. Каждая запись журнала расчетов характеризуется объектом, видом расчета, периодом действия, периодом регистрации, результатом и дополнительными реквизитами?
9. Важнейшими свойствами журнала расчетов является периодичность, справочник объектов и точность результата.
10. Количество дополнительных реквизитов в журналах расчетов неограниченно. Информация по ним вносится перед выполнением записи функциями `ВвестиРасчет()` и `ЗаписатьРасчет()`.
11. Журнал расчетов может иметь периодичность: день, неделя, месяц, квартал, год. Периодичность журнала расчетов менять нельзя, если он уже содержит записи. Если все-таки ее нужно поменять, то для этого нужно переименовать старый журнал и создать новый с новой периодичностью. Если требуемая по условиям задачи периодичность не предусмотрена программой (например, 10 дней), то можно задать периодичность большую, чем требуется, а при вводе записей расчетов указывать требуемую периодичность действия расчетов. Например, налоги с фонда оплаты в типовой конфигурации рассчитываются ежемесячно, а налоговый период — год.
12. При смене расчетного периода записи прошлого периода архивируются, и устанавливается новый период регистрации. Расчетный период можно устанавливать на несколько периодов вперед и на несколько периодов назад. Период журнала расчетов можно менять средствами встроенного языка.

Для добавления записей в журнал расчетов используются функции `ВвестиРасчет()` и `ЗаписатьРасчет()`.

Если атрибут "Сторно" журнала расчетов был установлен программным образом, то его можно программно переопределить.

Правила перерасчета предназначены для определения "зависимых" и "ведущих" видов расчета. Записи зависимых видов расчета должны быть пересчитаны при изменении записей ведущих видов расчета.

Установленные при конфигурировании правила перерасчета могут отключаться или переназначаться программными средствами.

Глава 12

Задания

12.1. Процедура импорта данных приведена в листинге П24.

Листинг П24. Импорт данных о сотрудниках

```
Процедура Сформировать()
Таб=СоздатьОбъект("Таблица");
Таб.ВывестиСекцию("Шапка");
Спр=СоздатьОбъект("Справочник.Сотрудники");
Подр=СоздатьОбъект("Справочник.Подразделения");
Спр.ИспользоватьДату(НачКвартала(ТекущаяДата()));
БД=СоздатьОбъект("XBase");
БД.ОткрытьФайл("d:\data.dbf");
Если БД.Открыта()=1 Тогда
        Если БД.Первая()=1 Тогда
        Пока 1=1 Цикл
                Код=БД.kod;
                Наименование=БД.Sotr;
                Оклад=БД.Oklad;
                Должность=БД.dolgnost;
                Подразделение=БД.Podr;
                Если Спр.НайтиПоКоду(Код)=0 Тогда
                        Спр.Новый();
                        Спр.Код=Код;
                        Спр.Наименование=Наименование;
```

```
                        Спр.Оклад=Оклад;
                        Спр.Должность=Должность;
                        Если Подр.НайтиПоНаименованию(Подразделение)=0 Тогда
                                Подр.Новый();
                                Подр.Наименование=Подразделение;
                                Подр.Записать();
                        КонецЕсли;
                        Спр.Подразделение=Подр.ТекущийЭлемент();
                        Спр.Записать();
                        Статус="Добавлен";
                Иначе
                        Статус="Пропущен";
                КонецЕсли;
                Таб.ВывестиСекцию("Строка");
                Если БД.Следующая()=0 Тогда
                    Прервать;
                КонецЕсли;
        КонецЦикла;
        КонецЕсли;
КонецЕсли;
Таб.Показать();
КонецПроцедуры
```

Предметный указатель

Ж

З

И

К

Л

М

О

П

Р

С

Т

Ф

Я