1С ПРОГРАММИРОВАНИЕ

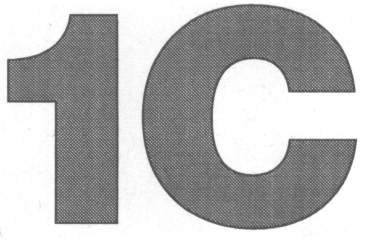
как дважды два

Знаете, как организовать свой бизнес?

- изучите язык «1С»
- научитесь говорить с бухгалтером на языке проводок и счетов
- изучите психологию общения с клиентами и подчиненными
- создайте свою клиентскую сеть
- наберите учеников и оттачивайте на них свое мастерство

С. Михайлов

С. Михайлов



программирование как дважды два

ООО «Тритон»

ББК 32.973.23-018.2я7
УДК 657.1.011(075)
М69

Михайлов С. Е.
М69 ^-программирование как дважды два. Самоучитель. — СПб.: Тритон, 2005. — 173, с: ил.

ISBN 5-94608-001-6

Книга основана на личном опыте предпринимательской деятельности автора в сфере 1С-программирования и должна обеспечить скорейшую адаптацию молодого специалиста на рынке бухгалтерского программного обеспечения. В ней, кроме примеров решения задач бухучета средствами языка «1С» и администрирования 1С-приложений, вы найдете практические советы, касающиеся организации собственного бизнеса, психологии общения с клиентами и заказчиками, проведения удаленного обучения через Интернет. Начинающим 1 С-программистам эта книга поможет освоить язык проводок и счетов, более опытным подскажет, как обойти сложности, связанные со спецификой использования счетов. На примеры конфигураций, описанных в книге, даны интернет-ссылка.

ББК32.973.23-018.2я7
УДК 657.1.011(075)

ISBN 5-94608-001-6

Краткое содержание

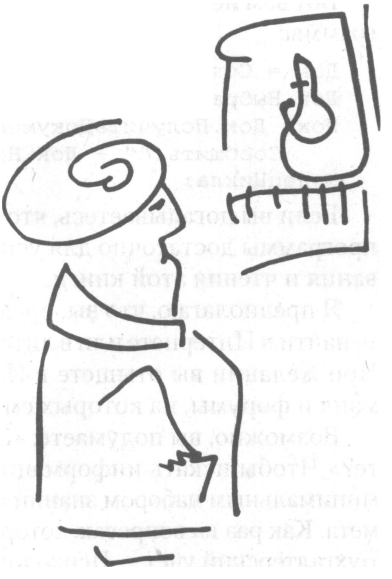
Содержание

Предисловие

Что должен уметь 1С-программист?

Программирование в области «1С» стоит на трех китах:

- навык программирования в среде «1С»,
- знание теории бухгалтерского учета,
- умение общаться, знание приемов психологического давления и защиты.



Уберите одного из «китов» — и из специалиста получится карикатура. Навык программирования не обсуждается: какой программист может считаться программистом, если он не умеет программировать? Без понимания теории бухгалтерского учета программист остается кодером, с которым не согласится работать ни один бухгалтер. А без знания элементарной психологии специалист обрекает себя на неинтересную рутинную работу.

Взгляд программиста на бухгалтерию отличается от взгляда бухгалтера. Чтобы определить, правильно ли бухгалтерское высказывание или нет, бухгалтер станет записывать столбцы проводок, в то время как программист нарисует граф. Интересно, что бухгалтер поймет рисунок программиста и программист поймет запись бухгалтера, но ни один, ни другой не перейдет на использование «чужого» языка. Это происходит потому, что бухгалтер говорит и мыслит на языке проводок, а программисту нужен взгляд «с высоты птичьего полета». Описание бухгалтерских схем с точки зрения программиста вы можете найти в этой книге.

Для успеха в программировании бухгалтерских систем важны умение программировать и знание предметной области, однако это не все навыки, которыми следует обладать. Как уже было сказано, еще одним важным (возможно, даже самым главным) умением для программиста является способность правильно взаимодействовать с заказчиком. Как сделать так, чтобы клиент платил, чтобы на программиста не вешали лишних работ, чтобы внедрение системы шло сверхбыстрыми темпами? Всё это вопросы психологической работы с клиентом, и им будет посвящена отдельная глава.

1С-программирование — это одна из областей, в которой можно вести бизнес без первоначальных вложений. Программист носит с собой весь свой «капитал» в голове, ему не нужны ни офисы, ни секретари. Как превратить возможность организовать собственное дело в реальность — одна из тем этой книги.

Минимальные требования к читателю

Чтобы научиться 1С-программированию, нужно иметь:

- желание,
- желание,
- желание,
- навыки программирования в принципе (которые, на мой взгляд, имеет каждый).

Вот вам небольшой тест. Попробуйте определить смысл написанной ниже программы:

```
Док = СоздатьОбъект("Документ.РасходнаяНакладная");
Док.ВыбратьДокументы();
Пока Док.ПолучитьДокумент() = 1 Цикл
    Сообщить("" + Док.НомерДок);
КонецЦикла;
```

Если вы догадываетесь, что делает эта программа, то вашего умения понимать программы достаточно для успешного саморазвития в области 1С-программирования и чтения этой книги.

Я предполагаю, что вы, предварительно или во время чтения книги, попробуете найти в Интернете или в литературе информацию по программированию в «1С». При желании вы отыщете в Интернете бесплатные справочники по системе команд и форумы, на которых сможете найти ответ на любой вопрос.

Возможно, вы подумаете: «Зачем мне книга, если все можно найти в Интернете?» Чтобы искать информацию, следует правильно задавать вопросы, обладать минимальным набором знаний в предметной области и понимать философию предмета. Как раз на вопросах, которые касаются философии «1С-Программирование — Бухгалтерский учет — Психология общения», я решил сделать упор в книге.

Я предполагаю, что вы уже прочитали или собираетесь прочитать пару книжек по бухгалтерскому учету.

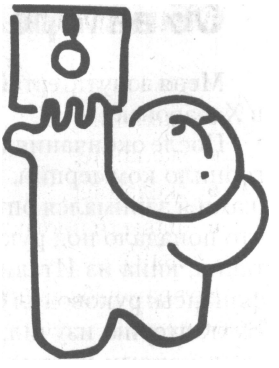
Желательно, чтобы у вас на компьютере были установлены программы «1 (^Бухгалтерия» и «1С:Торговля и Склад» или хотя бы одна из них. Изучать программирование без выполнения практических упражнений невозможно.

Я также надеюсь, что читатель будет амбициозно настроенным человеком. Часть книги посвящена строительству бизнеса в сфере 1С-программирования, и она вряд ли будет интересна замкнутым интровертам.

Структура книги

Эта книга является плодом восьмилетнего опыта работы с продуктами «1С» и бухгалтерским учетом, а также пятилетнего опыта предпринимательской деятельности. Она рассчитана прежде всего на студентов последних курсов, перед которыми стоит вопрос: в какой области применить свои силы после окончания учебы. Цель книги -- обеспечить скорейшую адаптацию молодого специалиста на рынке бухгалтерского программного обеспечения.

Книга состоит из восьми глав.



- В главе 1 приводится теория бухгалтерского учета с точки зрения программиста-практика. Изучив приведенные в нем понятия, вы освоите язык, на котором вам придется общаться с заказчиком.
- Глава 2 содержит информацию о принципах постановки задач и набор примеров, которые помогут вам продолжить самообразование в области 1С-программирования.
- Глава 3 содержит советы программисту: как организовать работу с клиентом, как формулировать задание на программирование, как поступить с тем, что было напрограммировано до вашего прихода к заказчику, как свести особенности схемы работы заказчика к стандартным схемам.
- Глава 4 описывает азы языка 1С-программирования: справочники, документы, таблицы значений, регистры, объект «Бухгалтерские итоги» и другие объекты языка «1С».
- Глава 5 содержит информацию о внутренней структуре базы данных, описывает механизмы лечения и устранения возможных сбоев при работе в «1С». Здесь описаны механизмы работы с журналом регистрации и проблемы, возникающие при работе задним числом.
- Глава 6 описывает приемы психологического нападения и защиты. Знание азов психологии позволяет на порядок увеличить эффективность общения с заказчиком.
- Глава 7 содержит рекомендации о том, как с нуля, без рубля в кармане, начать бизнес в области обслуживания компьютеров и программного обеспечения.
- Глава 8 описывает пример обучения 1С-программированию через Интернет.
- В приложениях находятся ссылки на архивы упомянутых в книге конфигураций и отчетов, примеры договоров на обслуживание программного обеспечения и пример оформления проспекта 1С-программиста.

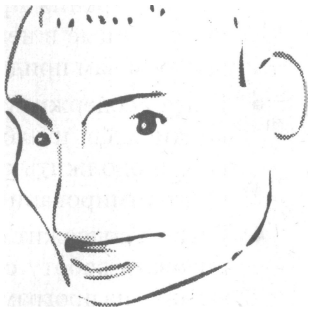
Используемые сокращения

Для краткости программы «1С:Бухгалтерия» и «1С:Торговля и Склад» я буду называть просто «Бухгалтерия» и «Торговля».

В книге также используются следующие сокращения:

БИ — бухгалтерские итоги,
МЦ — материальные ценности,
НДС — налог на добавленную стоимость,
ТА — точка актуальности итогов,
ТМЦ — товарно-материальные ценности.

Об авторе



Меня зовут Сергей Евгеньевич Михайлов. Я живу в Хабаровске.

После окончания института жизнь бросила меня в горнило коммерции. Вместе с двумя своими товарищами я занимался оптовой продажей. Продавали все, что попадало под руку: кондиционеры, продукты питания, вина из Италии. В нашей фирме я отвечал за финансы: руководил бухгалтерией, получал кредиты. Бухгалтерию изучил, прочитав пять книжек по бухгалтерскому учету. Именно тогда я в первый раз встретился с «1С» под DOS, которую переработал под складское хозяйство. Главное, что восхищало в «1С», — это то, что я мог изменять программу. Кроме того, среда предоставляла шаблоны и демонстрационные примеры.

Потом я попробовал организовать оптовый бизнес сам, затем работал 1С-программистом у дистрибьютора «Proctor and Gamble», и наконец, после этих жизненных перипетий, организовал свой 1С-бизнес, которым и занимаюсь до сих пор. Бизнес заключается в обслуживании 1С-программ и их адаптации под изменяющиеся нужды пользователей.

У меня нет сертификатов о том, что я являюсь специалистом в области 1С-программирования. Сначала я просто не предполагал, что такие сертификаты есть, а потом они мне стали нужны, как костыли здоровому человеку. Я не даю рекламу в средствах массовой информации: сначала на это не было денег, а потом я разработал схему, позволяющую находить клиентов без особых на то затрат. Сейчас среди моих клиентов — заводы и крупные торговые предприятия.

Благодарности

Благодарю Алиеву Татьяну Семеновну за редактирование бухгалтерской части; Илью Цветкова за читку материалов, а также как первого ученика, на котором я проверял материалы книги.

Благодарю редактора за ту долю перца, которую она насыпала на «сырые» с литературной точки зрения фрагменты книги.

О консультанте книги по вопросам администрирования

Отдельно хотелось бы поблагодарить Александра Черкова — за подготовку материалов по программированию и за консультации по вопросам администрирования и программирования.

Александр Анатольевич Черков занимается проектированием баз данных в области управления и учета. Его первой базой данных был АРМ (тогда это называлось «автоматизированным рабочим местом») учета времени преподавателей кафедры, на которой он работал лаборантом. Позже он участвовал в создании

баз данных в области ГИС и написании графических библиотек для работы с векторной графикой. Накопленный опыт обслуживания компьютеров, локальных сетей, бухгалтерских и торговых рабочих мест позволяет ему заниматься предпринимательской деятельностью в области комплексного обслуживания компьютерной техники и бухгалтерского программного обеспечения на основе «1С». Познакомился с «1С» четыре года назад. Считает, что «1С» лучше всего подходит для освоения программистами-новичками.

Глава 1
Теория бухгалтерского учета для программиста

— А вот эту цифру мы округляем.
— Понятно, а до какого разряда?
— При чем тут разряды? В кружок обводим!

Из разговора бухгалтера и программиста

В этой главе я приведу ряд вопросов, которые возникали у меня в ходе общения с бухгалтерами. В основном это вопросы оптовой и розничной торговли, а также производства. Вопросы учета основных средств здесь не рассматриваются, поскольку с ними практически не возникает проблем. Нет здесь и вопросов, связанных с заработной платой, так как это отдельная тема. Не рассматривается также процесс закрытия баланса: баланс — это забота бухгалтера, а не программиста. Не говорится и о взаимоотношениях с учредителями.

В главе приводятся коды программ и примеры их применения. Но хотелось бы предостеречь читателя от слепого использования программ и рекомендаций. Многие примеры и программы специально упрощены для учебного пособия. Многие бухгалтерские схемы являются верными лишь в первом приближении. Чтобы упрощенную схему сделать схемой, которая соответствует действительности, я делаю так: сначала объясняю клиенту свое «упрощенное» видение ситуации, а потом прошу довести схему до «особенностей предприятия».

Я не ставлю перед собой цель сделать вас бухгалтером, но, хочу дать вам в руки метод, который позволит находить с бухгалтерами общий язык.

Возможность разработки программ без предварительного точного описания — это плод практической работы и знания терминологии (в теории и на практике). Конечно, опыт работы в предметной области вам ничто не заменит, однако терминологию вы можете уяснить и не будучи бухгалтером.

Теория бухгалтерского учета

Основные понятия и обозначения

Счет — категория бухгалтерского учета («Касса», «Банк», «Товар в пути», «Товар на складе», «Товар в резерве»). Счета в бухгалтерском учете предназначе-

ны для текущего учета хозяйственных операций, для их обобщения и группировки по экономическому содержанию.

Субсчет — дополнительно детализированный счет.

Корреспонденция — связь между двумя счетами при записи хозяйственных операций.

Счета, участвующие в корреспонденции, называются *корреспондирующими счетами*.

Субконто — детализация счета (справочник, перечисление, документ). У счета 41 («Товары») может быть назначено субконто «Справочник Товары» и субконто «Справочник Склады».

Для записи проводки будет использоваться следующая конструкция:

Дата / Счет дебета {Субконто дебета} / Счет кредита {Субконто кредита} / Сумма.

Например, чтобы выполнить операцию выдачи денег из кассы подотчетному лицу, следует записать следующую проводку:

01.01.04 / 71 {Подотчетное лицо А} / 50 {Касса Б} / 100-00.

Когда дата проводки не важна, будет использоваться такая запись:

51 {Счет А} / 50 {Касса Б} / 100-00.

Когда важно заострить внимание только на сути бухгалтерских записей, а другая информация не важна, делается следующая запись:

51/50.

План счетов

Программисту нет необходимости помнить весь План счетов, однако важно ориентироваться в нем. План счетов можно найти в любой конфигурации «Бухгалтерии».

Ниже приведены счета, используемые в данной книге. Звездочкой помечены те, запоминание которых более необходимо по сравнению с остальными. Опыт показывает, что десятикратного повторения списка достаточно для прочного запоминания кодов счетов.

00 — Вспомогательный счет
01 — Основные средства
02 — Амортизация основных средств
10 — Материалы
16 — Отклонение в стоимости МЦ
19* — НДС по приобретенным ценностям
20 — Основное производство
41.1* — Товары на складах
41.2 — Товары в розничной торговле
41.3 — Тара под товаром и порожняя тара
42 — Торговая наценка
44 — Расходы на продажу

50* - Касса
51* — Расчетные счета
57 — Переводы в пути
60* — Расчеты с поставщиками
62.1* — Расчеты с поставщиками (в рублях)
62.2 — Авансы, выданные в рублях
68.2 - Налоги и сборы (НДС)
70* — Расчеты по оплате труда
71* — Расчеты с подотчетными лицами
90* — Продажи
90.1 —Выручка
99 — Прибыли и убытки

Счета используются для формирования проводок. Проводки объединяются в операции.

Операция — это набор логически связанных проводок, например проводок, формируемых документом, отражающим поступление материальных ценностей Если проводки одной операции разнести в разные места, то логика проводок будет потеряна и будет трудно понять, какой смысл имела каждая проводка.

Почему бухгалтерская запись делается так, а не иначе?

Бухгалтерские записи хозяйственных операций производятся в следующей форме:

Дата / Счет дебета / Счет кредита / Сумма / Пояснение.

Программист заметит, что приведенные выше записи похожи на строки базы данных.

Возможно, вы подумаете, что эта запись слишком сложна. Нельзя ли записать как-нибудь проще? Давайте попробуем разработать структуру базы для регистрации хозяйственных записей, например:

- Поступили деньги в *кассу* от *покупателя* 20.03.04 на сумму 200 р.
- Выданы деньги из *кассы* в *подотчет* 20.03.04 в сумме 100 р.
- Деньги с *подотчета* выплачены *Поставщику* интернет-услуг 20.03.04 в сумме 50 р.

Для начала приведем записи к однотипному виду, например **к** такому:

Дата / Получатель / Источник / Сумма / Пояснение.

Благодаря типизации записей их длина сократилась, а наглядность увеличилась:

- **20.03.04 / Касса / Покупатель / 200-00 /,**
- **20.03.04 / Подотчет / Касса / 100-00 /,**
- **20.03.04 / Поставщик / Подотчет / 50-00 / интернет-услуги.**

Сделаем следующий шаг: закодируем поля «Получатель» и «Источник» машинообразными шифрами:

- Подотчет — код 71,
- Покупатель — код 62,
- Касса — код 50,
- Поставщик — код 60.

Теперь записи можно представить в следующем виде:

- 20.03.04/50/62/200-00,
- 20.03.04/71/50/100-00,
- 20.03.04/60/71/50-00.

Сравните структуру придуманной нами записи с приведенной выше структурой бухгалтерской проводки. Правильно, они совпадают! С единственным различием, что в нашей записи поля именуются «Получатель» и «Источник», а в проводках — «Дебет» и «Кредит».

Бухгалтер прочитает операцию по выдаче подотчета работнику так: с кредита пятидесятого счета в дебет семьдесят первого счета списать сумму 100 р.

Словосочетание «с кредита счета» подразумевает уменьшение средств счета. Словосочетание «в дебет счета» обозначает их увеличение.

Как записать проводку в виде графа?

Представлю вышеприведенные проводки в виде графа (рис. 1.1).

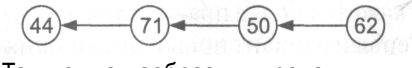


Рис. 1.1. Так можно изобразить проводку в виде графа

При составлении графа введем условие: счет-источник писать справа, а счет-получатель — слева, то есть стрелки должны следовать справа налево. Такой граф будет легко перевести на язык проводок, так как в проводке сначала пишется получатель (дебет), а затем — источник (кредит).

Узел графа (рис. 1.2), он же — бухгалтерский счет, описывается тремя параметрами: *текущим состоянием, дебетовым оборотом (ДО)* и *кредитовым оборотом (КО)*.

Рис. 1.2. Узел графа соответствует бухгалтерскому счету

Во времени его можно описать математическим языком с помощью шести параметров (рис. 1.3):

- СНД — сальдо начальное дебетовое,
- СНК — сальдо начальное кредитовое,
- ДО — дебетовый оборот,
- КО — кредитовый оборот,
- СКД — сальдо конечное дебетовое,
- СКК — сальдо конечное кредитовое.

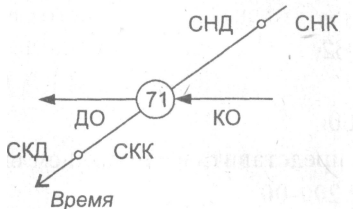


Рис. 1.3. Добавив временную ось, разместим на ней параметры проводки

Все эти параметры предоставляются программисту в «Бухгалтерии».

Как правильно записать проводку?

Еще раз вернемся к анализу формы бухгалтерских проводок по подотчету и рассмотрим два варианта их записи.

Вариант 1 — бухгалтерская запись:

- 01.01.04 / 41 / 60 / 100-00 (приход товара от поставщика),
- 10.02.04 /'90 / 41 / 100-00 (расход товара покупателю).

Вариант 2 — записи «здравого смысла»:

- 01.01.04 / Т / +100-00 (приход товара),
- 10.02.04 / Т / -100-00 (расход товара).

Хотя первая запись кажется более правильной, в бухгалтерском учете используются оба варианта. Первый вариант проводок выполняется для балансовых счетов, а второй вариант используется для забалансовых счетов.

Типы счетов

Счета могут быть активными и пассивными, балансовыми и забалансовыми, отражать количественный и валютный учет.

Активные и пассивные счета

Счета бухгалтерского учета по-разному воспринимают ситуацию, когда на счете образуется положительный или отрицательный остаток. Некоторые счета могут быть только отрицательными, а другие должны быть только положительными. Например, касса может быть только положительной, а уставной капитал (счет 80) — только отрицательным.

Когда счет должен содержать только положительный остаток, по-бухгалтерски он называется *активным счетом*. Когда счет может быть только отрицательным, он называется *пассивным*. Когда счет может быть как отрицательным, так и положительным, он называется *активно-пассивным*.

ПРИМЕЧАНИЕ Поясню, в чем смысл отрицательных (пассивных) счетов, на примере уставного капитала. Чтобы предприятие начало работать, в него необходимо вложить деньги. Этот начальный вклад делается со счета «Уставной капитал», например, следующей проводкой: Касса / Уставной капитал / 1 000 000-00. После этой операции на счете «Касса» появится положительная сумма +1 000 000-00, а на счете «Уставной капитал» — отрицательная сумма -1 000 000-00.

Счет 62.2 — пассивный. Это значит, что у него остатки должны быть отрицательными (кредитовыми). Если у пассивного счета образовался положительный остаток, то он должен записываться в кредитовой части баланса с минусом.

Счет 50 («Касса») является активным (дебетовым) счетом. Если на нем будет отрицательный остаток, то это будет ошибкой (представьте «отрицательные» деньги у себя в кармане). Как и остаток наличных денег, остаток товара на сладах может быть только положительный.

Как же разобраться, активный счет или пассивный? Все просто. Если он похож на кассу (счет 50), то это активный счет. Если он похож на уставной капитал (счет 80), то он пассивный.

Программисту совсем не обязательно знать, какой счет активный, а какой пассивный. В своей практике я ни разу не встречал каких-либо особенностей написания проводок в зависимости от того, является ли счет активным или пассивным.

Балансовые и забалансовые счета

Балансовый счет увеличивается только в результате уменьшения какого-либо другого счета. Балансовые счета образуют замкнутую цепочку (рис. 1.4). Сумма приходов на каждый узел (счет) такой цепочки будет равна сумме расходов на каждый узел (счет) цепочки.

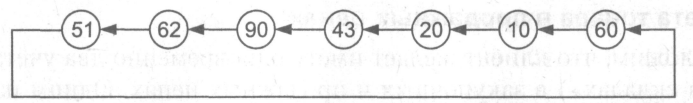


Рис. 1.4. Балансовые счета корреспондируют с другими счетами и образуют замкнутые цепочки

Забалансовый счет не может выстраиваться в замкнутую цепочку (рис. 1.5).

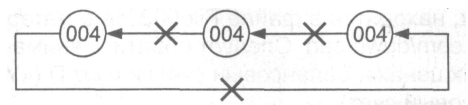


Рис. 1.5. Забалансовые счета не могут корреспондировать с другими счетами

Балансовый счет при формировании проводки обязательно должен иметь счет-корреспондент. Забалансовый счет не может содержать счет-корреспондент. Забалансовый счет не участвует в двойной записи, не требует в проводках наличия корреспонденции и не может корреспондировать с балансовыми счетами. Например, балансовый счет 50 («Касса») может делать такую проводку:

50 / 51 / 100-00 (поступление денег из банка в кассу),

а забалансовый счет 004 («Товары на ответственном хранении») может участвовать в следующих проводках:

004 / / 100-00 (поступление ТМЦ на ответственное хранение),
/ 004 / 100-00 (снятие ТМЦ с ответственного хранения).

Забалансовые счета могут использоваться для расширения аналитики, предусмотренной балансовыми счетами, например, когда требуется учитывать находящуюся у клиента тару (кеги).

Аналогом забалансового счета в «Торговле» является *регистр*. Аналогов балансовых счетов в «Торговле» не предусмотрено.

Пример использования забалансового счета при учете возвратной тары

Требуется ввести учет возвратной тары, отданной клиенту.

Так выглядит традиционная проводка при отгрузке товара (счет 41) покупателю (счет 62):

90 {Продажи} / 41 {Номенклатура},
62 {Клиенты} / 90 {Продажи}.

Вариант 1. Ввести субконто «Номенклатура» к счету 62. Данное решение не совсем красивое, поскольку загромождает счет 62 лишней аналитикой.

Вариант 2. Ввести в План счетов забалансовый счет «ТАР», к которому прикрепить субконто «Номенклатура» и «Контрагент», и разрешить по счету количественный учет.

ПРИМЕЧАНИЕ Конфигурация, реализующая учет возвратной тары при помощи забалансового счета, находится в файле File001.zip в материалах к данной книге по адресу www.piter.com/download. Следует обратить внимание на забалансовый счет ТАР.

Пример использования забалансового счета для учета товара в продажных ценах

Предположим, что клиент желает иметь одновременно два учета по счету 41 («Товар на складах») в закупочных и продажных ценах. Одним из возможных решений этой проблемы может быть внесение в План счетов забалансового счета, который выполнял бы ту же функцию, что и счет 41, но делал бы проводки в продажных ценах.

ПРИМЕЧАНИЕ Конфигурация, использующая забалансовый счет для учета товара в продажных ценах, находится в файле File002.zip в материалах к данной книге по адресу www.piter.com/download. Следует обратить внимание на счет 41 («Учет товара в закупочных ценах», балансовый счет) и счет П («Учет товара в продажных ценах», забалансовый счет).

Валютный учет

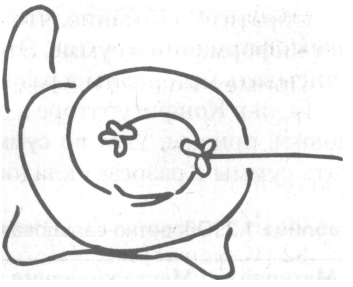
Валюта — это такой объект учета, который в зависимости от времени меняет свою стоимость. Номинал валюты остается постоянным, а рублевый эквивалент изменяется в зависимости от курса валюты.

Что делать, если аналитики стандартного Плана счетов не хватает?

Как вы думаете, можно ли придумать какой-либо учет, кроме суммового, количественного и валютного (стандартных измерений Плана счетов)?

Например: необходимо организовать учет на свиноводческой ферме. Вам сообщают, что требуется учитывать свиней поголовно, в килограммах и в рублях.

Если немного поразмыслить, то для учета свиней получится следующая схема:



- учет свиней в килограммах — количественный учет;
- учет свиней в рублях — суммовый учет;
- поголовный учет — некое недостающее измерение.

Решение, которое напрашивается само собой, — ввести дополнительное измерение «Головы» — не оптимально. Я вас специально запутал. Задачу можно решить без привлечения необычных средств, но для этого необходимы уточняющие данные. Для начала попробуйте переформулировать вопрос в более понятную для вас форму. Для решения задачи я бы задал следующий вопрос: «Будьте добры, уточните, пожалуйста: если каждая отдельная свинья учитывается в килограммах, то значит ли это, что свинья должна идентифицироваться относительно других и иметь инвентарный номер?»

Скорее всего, заказчик ответит, что инвентарные номера свиньям не присваиваются, а поголовный учет требуется, например, чтобы определять факт кражи. Такую задачу можно решить введением забалансового счета, на котором бы учитывалось поголовье свиней, то есть задача решается обычными методами.

Вариант решения задачи:

Введем в План счетов счет 11 («Животные на выкорме»). Количественный учет на этом счете будет вестись в разрезе веса животных.

Введем забалансовый счет СВИ. Количественный учет на этом счете будет вестись в разрезе голов животных.

Количественный учет субконто

Если вы посмотрите в типовом Конфигураторе субконто Места хранения на счете 10.1, то в его свойствах не будет установлен флажок Учет по сумме. (Чтобы проверить это, откройте в Конфигураторе План счетов, установите курсор на субконто Материал счета 10.1 и два раза щелкните мышью.)

Оборотно-сальдовая ведомость в разрезе материалов по счету 10.1 будет следующей (табл. 1.1).

Таблица 1.1. Оборотно-сальдовая ведомость по счету 10.1 в разрезе материалов

Материалы; Места хранения за IV квартал 2003 г.						
Субконто	**Сальдо на начало периода**		**Обороты за период**		**Сальдо на конец периода**	
	Дебет	**Кредит**	**Дебет**	**Кредит**	**Дебет**	**Кредит**
Бумага Количество			100.00 10.000		100.00 10.000	
Основной склад Количество			 10.000		 10.000	
Итого развернутое **Итого**			 **100.00**		**100.00** **100.00**	

Обратите внимание, что в оборотно-сальдовой ведомости в разрезе складов нет информации о сумме. Это может быть препятствием в случае, если вы решите списывать материалы в разрезе складов.

Если в Конфигураторе в свойствах субконто МестаХранения счета 10.1 установить признак Учет по сумме, то оборотно-сальдовая ведомость будет показывать суммы в разрезе складов (табл. 1.2).

Таблица 1.2. Оборотно-сальдовая ведомость по счету 10.1 в разрезе складов

Материалы; Места хранения за IV квартал 2003 г.						
Субконто	**Сальдо на начало периода**		**Обороты за период**		**Сальдо на конец периода**	
	Дебет	**Кредит**	**Дебет**	**Кредит**	**Дебет**	**Кредит**
Бумага Количество			**100.00** **10.000**		**100.00** **10.000**	
Основной склад Количество			**100.00** **10.000**		**100.00** **10.000**	
Итого развернутое **Итого**			 **100.00**		**100.00** **100.00**	

Стандартная конфигурация «Бухгалтерии» списывает материалы (счет 10) по одной для всех складов себестоимости. Если вы решите переделать схему списания (чтобы для каждого склада была своя себестоимость), то вам будет необходимо установить в свойствах субконто Места хранения счета 10 флажок Учет по сумме.

Практические вопросы использования бухгалтерских счетов

Как убрать «пересортицу» между счетами?

Предположим, что в бухгалтерии ведется учет по многим субсчетам одного счета, например:

- 60.1.1 — Поставщики материалов,
- 60.1.2 — Поставщики тары,
- 60.1.3 — Поставщики топлива, и т. д.

Теперь предположим, что один и тот же поставщик является поставщиком и материалов, и тары. Тогда с течением времени между субсчетами образуется пересортица, например такая:

Клиент А, счет-остаток:

60.1.1--100-00,
60.1.2-+100-00,
60.1.3-0-00,
60.1.4-0-00.

Клиент ничего не должен, если делать анализ счета 60.1, и должен 100 р., если делать отчет по счету 60.1.2.

Первый вариант решения — сделать проводку:

60.1.1/60.1.2/+100-00. (1)

Второй вариант решения — сделать две проводки:

00/60.1.2/-100-00, **(2)**

00/60.1.1/+100-00. (3)

Этот вариант решения проблемы представляется более предпочтительным, поскольку проводка (1) попадет в акт взаиморасчетов и увеличит обороты дебета и кредита, а проводки (2) и (3) не дадут увеличения дебетовой и кредитовой частей баланса, в то же время сальдо по счету 60 останется неизменным.

Вышеописанная схема позволяет оставлять документы такими, какие они были сформированы (то есть не исправлять первичные документы). Для решения подобных проблем можно сделать обработку, которая создавала бы одну консолидированную операцию.

ПРИМЕЧАНИЕ Обработка «Сбор мусора 62», решающая проблему пересортицы между субсчетами счета 62, находится в файле File003.zip в материалах к данной книге по адресу www.piter.com/download.

Как исправить ошибки, не исправляя их?

Допустим, вы закрыли баланс и на счете 10 («Материалы») был остаток 100 рублей. На следующий день был перепроверен один из документов, списывающий материалы по средней стоимости. В результате этого остаток на счете 10 изменился и стал равным 101 р., что привело к нарушению баланса. Что делать с разницей в один рубль? Нужно каким-то образом вернуть 100 р. на место.

Исправление можно провести следующими проводками:

31.01.03/20/10/+1-00,
01.02.03/20/10/-1-00.

Таким образом вы сделали сумму такой, как вам было нужно, и «не сделали изменений», вернув правильное значение счета.

ПРИМЕЧАНИЕ Во время чтения данного раздела один из проверяющих бухгалтеров-профессионалов отметил, что способ «исправления не исправляя» посредством манипулирования датами совершенно недопустим. Доводы были следующими. Во-первых, если произошло изменение баланса, то затрагивается не один счет, а несколько, следовательно, исправлять надо несколько счетов. Во-вторых, следует выяснить правомерность изменений (в каком случае было верно, до изменения данных или после) и после этого привести баланс в верное состояние. Для выяснения сути ошибки предлагалось сверить текущие компьютерные остатки с распечатками. Чтобы не встречаться с ситуацией, когда после перепроведения документа баланс рушится, следует в конце месяца препроводить все документы.

Операции, продленные во времени

Под операцией, продленной во времени, будем понимать такую операцию, которая начинается сегодня, а завершается через неопределенное время, но точно

не сегодня. Например, перечисляются деньги с одного расчетного счета организации на другой расчетный счет этой же организации.

Делать такую проводку:

01.01.04 / 51 {Расчетный счет Б} / 51 {Расчетный счет А} / 100-00

нельзя, поскольку деньги с расчетного счета «А» могут уйти сегодня, а поступить на расчетный счет «Б» только завтра. Кроме того, для каждого расчетного счета заполняется свой документ банковской выписки, то есть вроде бы должны составляться две операции на один расчетный счет за разными числами:

01.01.04 / 51 {Расчетный счет Б} / 51 {Расчетный счет А} / 100-00,
02.01.04 / 51 {Расчетный счет Б} / 51 {Расчетный счет А} / 100-00.

Однако и эти проводки неверные, поскольку обороты по счетам завышаются в два раза.

Проблема продления операции во времени решается введением еще одного счета, например счета 57 («Переводы в пути»). Теперь все встает на свои места:

01.01.04 / 57 / 51 {Расчетный счет А} / 100-00,
02.01.04 / 51 {Расчетный счет Б} / 57 / 100-00.

Распознавать ситуацию, в которой нужно использовать счет, аналогичный счету «Переводы в пути», можно с помощью следующих вопросов:

- Я правильно понял, что бухгалтерская операция одна, но начинается эта операция в один день, а заканчивается следующим днем?
- А эта ситуация случайно не похожа на случай использования счета «Переводы в пути»?
- Вы хотите сказать, что следует делать как бы две проводки, когда на самом деле операция должна быть одна?

Ошибки при перемещении ТМЦ между складами

При использовании компоненты «Распределенная информационная база данных» журнал, в который заносятся изменения документов, не переносится вместе с базой данных, поэтому может быть затруднен поиск лиц, сделавших в документах несанкционированные изменения.

Допустим, была допущена пересортица во время перемещения товаров между складами одной и той же фирмы и оператор на принимающем складе (Оператор 2) решил исправить ошибки. После того как произойдет обмен между информационными базами, документ, измененный Оператором 2, поступит в базу данных оператора склада-отправителя (Оператор 1) и создаст ошибку, так как с точки зрения Оператора 1 данные в документе изменились «самостоятельно», ведь он не знает, что Оператор 2 внес изменения в документ (рис. 1.6).

Как разобраться, кто изменил документ, является ли изменение в документе ошибкой программы, сбоем в базе данных, или изменение было введено преднамеренно? Для этого составляют компьютерные распечатки движения товара и сравнивают их с первичными документами, что отнимает много времени.

Какие есть варианты решения проблемы?

Общее правило: запретить изменять документы чужого автора.

Следствия из правила:

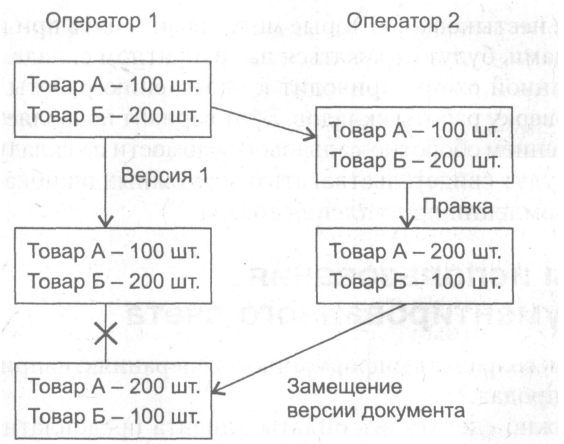


Рис. 1.6. Иллюстрация «самопроизвольного» изменения данных

1. Перемещение может формироваться только на одном из складов. На другой склад перемещение должно поступать вместе с обновлением базы данных, то есть база данных сначала попадает в центральную базу данных, а затем поступает во второй филиал.
2. Товар следует оприходовать как есть, а на сумму излишков и недостачи формировать документы поступления товара и списания.
3. Вместо перемещения делать расходную накладную на складе-источнике и приходную накладную на складе-приемнике.
4. Оформить документ перемещения, но при этом использовать склад-буфер. Склад-источник должен перемещать товар на склад-буфер, а не на склад-приемник, а склад-приемник должен получать товар не со склада-отправителя, а со склада-буфера (рис. 1.7).

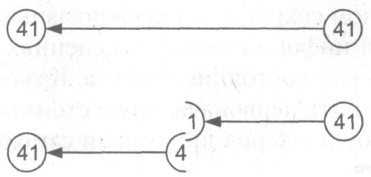


Рис. 1.7. Использование склада-буфера позволяет избежать «самопроизвольного» изменения данных и разделить во времени операции расходования и приходования товаров

Как использовать склад-буфер при перемещении?

Обычная проводка выглядит так:

41 {Склад 1} / 41 {Склад 2}.

Проводки с использованием склада-транзита выглядят так:

- **41 {Склад-транзит} / 41 {Склад 2}** — эту проводку делает склад 2;
- **41 {Склад 1} / 41 {Склад-транзит}** — эту проводку делает склад 1.

Все товарные нестыковки, которые могут возникнуть при перемещении товаров между складами, будут отражаться на транзитном складе.

Внедрение данной схемы приводит к облегчению работы бухгалтера, ответственного за проверку работы складов. Этот вариант позволяет находить ошибки простым составлением оборотно-сальдовой ведомости по складу-транзиту: остатки на этом складе будут свидетельствовать о возможных ошибках, пересортице, неправильном оформлении поступления товара.

Примеры использования самодокументированного счета

Пример 1. Как сохранять информацию об операциях, например о записях, сделанных в книгу продаж?

Для этого можно сделать все оплаты клиента предоплатными: оплата будет поступать на счет 62.2, а после формирования книги продаж те оплаты, которые должны в нее войти, должны будут сформировать проводку **62.1 / 62.2.**

Пример 2. Как сделать так, чтобы изменения во взаиморасчетах нельзя было бы произвести без главного бухгалтера?

1. Пусть бухгалтер-взаиморасчетчик делает проводки **50 / 57 {Клиент}.**
2. Создать программу, которая делает проводки **57 {Клиент} / 62 {Клиент}** и автоматически закрывает счет 57. Пусть права на запуск этой программы будут даны только главному бухгалтеру. Теперь вся работа задним числом будет отражаться на счете 57.

Сохранение информации при помощи Плана счетов

Знаете ли вы, что суммы по балансовой стоимости основных средств хранятся на одном счете (01), а износ основных средств хранится на другом счете (02)? Оказывается, таким образом сохраняется первоначальная информация о состоянии основного средства и информация об изменениях (амортизации). Разница между счетами — это текущее состояние объекта. Казалось бы, можно было сделать проще, например хранить первоначальную стоимость в карточке основного счета, однако поколения бухгалтеров придумали самодокументирующуюся схему учета основных средств.

Такое свойство можно использовать для отслеживания изменений, сделанных определенным пользователем системы. Допустим, с целью сохранения первоначальной информации о выручке, поступившей в кассу, можно реализовать следующую схему проводок:

- **О.1 / 50 / 100-00** — проводка должна формироваться кассиром;
- **62 / О.1 / 100-00** — проводка должна делаться бухгалтером.

Счет О.1 следует предварительно ввести в План счетов.

Теперь все несанкционированные изменения в закрытом периоде будут шиты белыми нитками, поскольку простейший анализ счета О.1 будет показывать изменения. Внедрение подобных схем сделает поиск ошибок и преднамеренных изменений максимально упрощенным.

Отрицательная проводка, или проводка задом наперед

Под *отрицательной* проводкой будем понимать проводку, сумма которой отрицательна:

счет А / счет В / - сумма.

Под *обратной* проводкой будем понимать такую проводку, сумма которой положительна, а корреспондирующие счета поменяны местами:

счет В / счет А / + сумма.

Проводка счет А / счет В / + сумма идентична проводке счет В / счет А / - сумма (рис. 1.8).

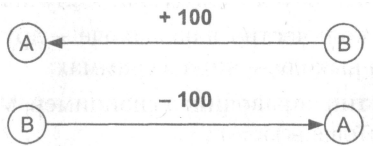
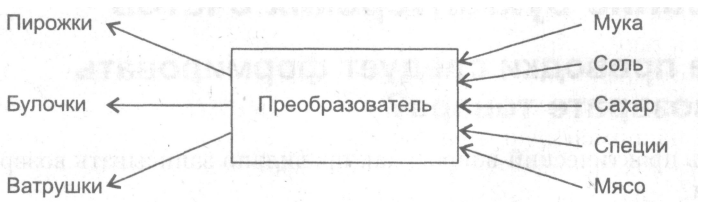


Рис. 1.8. Эти проводки идентичны

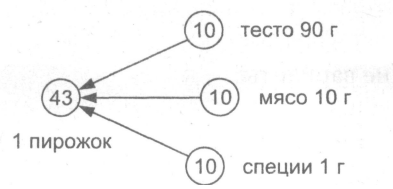
Применение мультиплексора в «Бухгалтерии»

Мультиплексор — это преобразователь многих входов во многие выходы. Аналогом мультиплексора является телефонная станция, которая позволяет соединить множество входящих звонков со множеством абонентов.

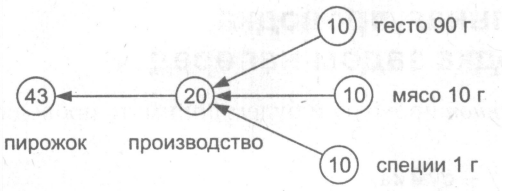
Мультиплексоры широко применяются в бухгалтерском учете. Например, на входе — материалы, а на выходе — товары. Если математически записать это выражение, то получается так: Товары = F (Материалы), то есть множество Товары является функцией от множества Материалы:



На бухгалтерском языке эту схему можно записать так:



Однако если записать эту операцию в виде проводок, получится противоречие: проводка, содержащая счет 43, должна быть одна, а проводок, содержащих счет 10, должно быть три. Выход прост: использовать мультиплексор — счет 20 («Производство»):



Проводки будут выглядеть так:

20 (производство) <— 10 (мясо 10 г),
20 (производство) <— 10 (тесто 90 г),
20 (производство) <— 10 (специи 1 г),
43 (1 пирожок) <*г*- 20 (производство).

Мультиплексоры применяются в следующих случаях:

- если на входе одно количество, а на выходе — другое, например на входе — штучные яйца, а на выходе — яйца в граммах;
- если на входе один тип справочника (например, Материалы), а на выходе — другой (например, Товары).

Определив, что в рассматриваемой вами схеме должен использоваться мультиплексор, вы должны задаться вопросом о бухгалтерском счете, через который движется информация. Например, в производстве таким счетом является счет 20 («Основное производство»). В случае, когда требовалось превратить ящики в граммы, я использовал счет 16 («Отклонения в стоимости МЦ»).

ПРИМЕЧАНИЕ Конфигурация, реализующая вышеописанную схему, находится в файле Fil6004.zip в материалах к данной книге по адресу www.pjter.com/download. Следует обратить внимание на документ «Требование-накладная». Документ превращает товары (счет 41) в материалы (счет 10), преобразуя при этом количества.

Описание бухгалтерских счетов

Какие проводки следует формировать при возврате товара?

А теперь практический вопрос: как правильно записывать возврат товара от покупателя?

Напомню, что отгрузка товара состоит из следующих проводок:

- 90/41/+,
- 62/90/+.

Возможны следующие варианты:

1. 90/41/-,
   62/90/-;
2. 41/90/+,
   90 / 62 / +;
3. 41/90/+,
   62 / 90 / -;

4. 90/41/-,
   90/62/+.

Какой из этих четырех вариантов предпочтительнее? Для того чтобы ответить на этот вопрос, следует определить лиц, заинтересованных в процессе возврата. Заинтересованными сторонами являются:

- кладовщик (счет 41),
- отдел реализации (начисление НДС),
- торговый агент или покупатель (счет 62).

Для кладовщика проводки 41 / 90 / +100-00 (1) и 90 / 41 / -100-00 (2) не идентичны. *Возврат* для кладовщика — это *приход,* такой же, как и приход от поставщиков, то есть возврат должен записываться в приходную (дебетовую) часть баланса. Если возврат записать как «расход с минусом» (то есть «не-расход»), то возникнут проблемы. Ведь кладовщик точно знает, что он списал со склада 100 единиц товара и затем получил обратно 20 единиц:

Документ	Приход	Расход
Расходная накладная	+ 100	
Возвратная накладная	+20	
Итого:	+20	+ 100

а не списал 80 единиц:

Документ	Приход	Расход
Расходная накладная	+ 100	
Возвратная накладная	-20 (не-расход)	
Итого:	0	+80

В первом случае получается расход 100 и приход 20, а во втором — только расход 80.

Поэтому возвратную проводку с точки зрения кладовщика следует писать, как указано в проводке (1).

С другой стороны, бухгалтер будет настаивать на том, чтобы записывалась проводка (2), потому что возвратная накладная должна уменьшать оборот счета 90.

Теперь рассмотрим проводки задолженности.

При оформлении возврата задолженности возможны следующие проводки:

- 90/62/+100-00, (3)
- 62/90/-100-00. (4)

Эти проводки идентичны, однако проводка (4) не завышает реализацию. То есть можно записывать как проводку (3), так и проводку (4).

ПРИМЕЧАНИЕ Как быть, если возникнут проблемы возврата на практике? При постановке задачи следует уточнять, что имеется в виду, когда речь идет о возврате, например, следующим вопросом: «Скажите, что вы имеете в виду, когда говорите о возврате: возврат — это поступление или возврат — это «не-расход»? Следует ли писать проводки с минусом, или они должны быть обратными проводками?»

Производство

Может быть, вы считаете, что производство — штука более сложная, чем торговля? Сейчас вы убедитесь, что это не так. Производственные проводки можно представить следующим графом (рис. 1.9):

- 20 / 10 — списание материалов в производство,
- 20 / 23 — списание вспомогательного производства,
- 20 / 26 — списание общехозяйственных расходов,
- 20 / 29 — списание обслуживающего производства и хозяйства,
- 43 / 20 — выпуск продукции.

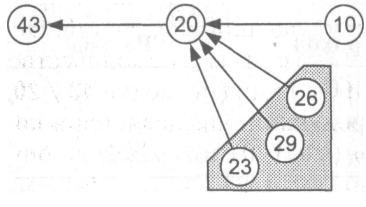


Рис. 1.9. Схема проводок производства с учетом счетов 23, 29 и 26

Все проводки можно разбить на следующие группы:

- 20 / 10 — передача материалов в производство (проводки делаются в момент передачи материала в производство);
- 20 / 2* — списание расходов (проводки делаются в конце месяца, когда будут произведены все расходы);
- 43 / 20 — выпуск продукции (проводки делаются в процессе выпуска готовой продукции).

Если заранее известно, какой материал идет на производство определенной продукции, то проводки 20 / 10 и 43 / 20 можно делать одним документом. В этом случае, если бы не было проводок по списанию расходов (20/2*), то себестоимость продукции определялась бы тем, какие материалы были списаны на производство конкретной продукции. Например, если на производство пятидесяти пирожков было потрачено муки на сумму 90 р. и моркови на сумму 10 р., то себестоимость одного пирожка определялась бы так: (90 + 10) / 50 = 2 р. за один пирожок.

Однако кроме материалов, производству требуются электроэнергия, затраты на заработную плату, отопление (рис. 1.10). Сумма дополнительных расходов становится известна только в конце месяца, поэтому и точная себестоимость продукции определяется только в конце месяца.

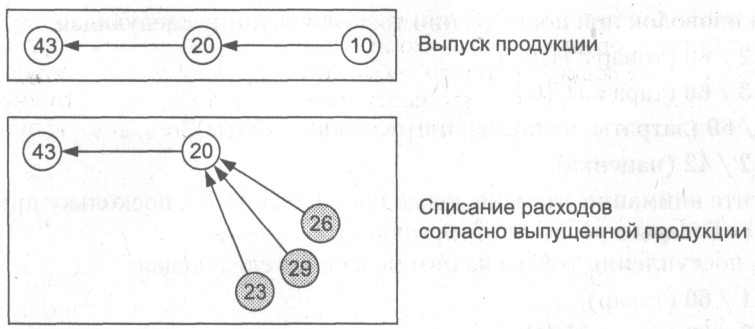


Рис. 1.10. Себестоимость продукции складывается из двух частей: материалов и списания расходов

В течение месяца производственное предприятие работает «вслепую», поэтому не всегда может своевременно отреагировать на резкое повышение затрат и перестроить свою работу, например отказаться от убыточного направления.

Списание расходов согласно выпущенной продукции — это расчет, который выполняется в конце месяца. Расчет формирует проводки 43 / 20 на рублевую добавку согласно объему выпущенной продукции. Обратите внимание, что проводка 43 / 20 учитывает только суммы, а не количество. Количественные проводки сделаны при выпуске продукции (проводки 43 / 20, 20 / 10)

Если себестоимость продукции определяется в конце месяца, то и списание себестоимости продукции (счет 90) тоже должно определяться только в конце месяца. Это значит, что проводки 90 / 43 в течение месяца можно формировать в количественном выражении. А в конце месяца должна выполняться программа, которая формирует суммовые проводки 90 / 43.

На рис. 1.11 белым цветом обозначены операции, выполняющиеся в течение месяца. Серым цветом помечены операции, которые выполняются в конце месяца.

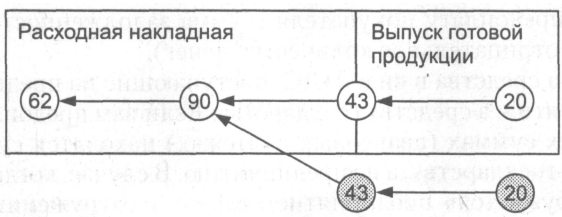


Рис. 1.11. Серым цветом обозначены проводки, выполняемые в конце месяца

ПРИМЕЧАНИЕ Конфигурация, реализующая работу со счетом 20, находится в файле File005.zip в материалах к данной книге по адресу www.piter.com/download.

Учет возвратной тары

Возвратная тара — это тот «товар», который должен вернуться назад. При поступлении на склад такой «товар» не наценивается.

Схема проводок при поступлении товара в розницу следующая:

- **41.2 /60** (товар с НДС),
- **41.3/60** (тара с НДС),
- **44/60** (затраты, например погрузочные работы),
- **41.2 / 42** (наценка).

Обратите внимание, что я не делаю проводок с НДС, поскольку приходую на счет 41.2 («Товары в розничной торговле»).

Схема поступлении товара на оптовый склад следующая:

- **41.1/60** (товар),
- **41.3/60** (тара с НДС),
- **44/60** (затраты, например погрузочные работы),
- 19/60 (НДС).

Когда тара возвращается поставщику, то делается проводка 60 / **41.3.**

Счет 62

Счет 62 («Задолженность покупателей») разбит на два субсчета — 62.1 («Счет послеоплаты») и 62.2 («Счет предоплаты»). Послеоплатный счет — активный и на нем хранится вся сумма задолженности предприятия. Предоплатный счет — пассивный и на нем хранятся суммы без НДС. Поэтому, кстати, если покупатель предоплатил, то становится затруднительно определить, на какую сумму следует произвести отгрузку (ведь сумма задолженности хранится без НДС!).

Как вы думаете, почему введена такая путаница с НДС? Какая разница, как учитывать суммы задолженности — с НДС или без НДС? Ведь речь идет всего лишь о математическом знаке задолженности. Допустим, покупатели А и Б не имели задолженности перед предприятием. Операция **62 {Покупатель А} / 90 / 100-00** образует задолженность на счете 62 (сумма задолженности положительная: на счете 62 стало больше денег, чем было). Операция **51 / 62 {Покупатель Б} / 100-00** образует предоплату покупателя (сумма задолженности отрицательная: на счете 62 стало отрицательное количество денег).

Дело в том, что средства в виде НДС, поступающие на предприятие, — это не средства предприятия, а средства государства. Если вам предоплатили деньги, то в предоплаченных суммах (авансовых платежах) находятся суммы НДС, которые принадлежат государству, а не предприятию. В случае, когда идет послеоплата, то НДС «отгружается» предприятием вместе с отгруженным по накладной товаром (то есть вы передаете обязательство по оплате НДС покупателю) и баланс не нарушается.

Поскольку НДС платится по итогам месяца, то и предоплата будет интересовать налоговые органы только на конец месяца, то есть если предоплата закрылась в течение месяца, то ее можно не показывать как предоплату.

На крупных предприятиях часто существует должность бухгалтера-взаиморасчетчика, который гоняет деньги между предоплатным и послеоплатным счетами. Иногда эти операции делаются вручную, а чем больше ручных операций, тем больше ошибок.

Хорошо было бы сократить до минимума ошибки, например, при помощи специальной обработки, которая бы самостоятельно определяла, каким образом следу-

ет формировать счета 62.1 и 62.2. Такие рассуждения встречают серьезные протесты со стороны бухгалтеров. Не потому ли, что они часть своей жизни потратили на бесполезную работу, которая может быть решена простой обработкой?

Главная сложность при работе со счетом 62.2 заключается в том, что на этом счете задолженность хранится без суммы НДС, поэтому просто сравнивать счета 62.1 и 62.2 нельзя (если на счете 62.1 образовался отрицательный остаток, то эту сумму нужно отправить на счет 62.2). Поэтому сначала следует восстановить информацию об НДС сторнированием проводок, сделанных в прошлом месяце:

- **62.1** / **62.2** / минус сумма с НДС на сумму кредитного остатка на счете 62.1, который был в прошлом месяце;
- **62.2** / **68.2** / минус сумма НДС на сумму кредитного остатка на счете 62.1, который был в прошлом месяце.

Если нужно реализовать предоплату, то следует сделать следующие проводки:

- 90 / **68.2** — реализация товара, по которому была предоплата;
- **62.2** / **68.2** — сторная проводка на сумму НДС с аванса.

Теперь, если на счете 62.1 образовался кредитовый (отрицательный остаток), то на сумму минуса следует сделать прямые проводки:

- **62.1** / **62.2** / сумма с НДС на сумму кредитового остатка на счете 62.1;
- **62.2** / **68.2** / сумма НДС на сумму кредитового остатка на счете 62.1.

Что значит «закрыть счет 90»?

В течение месяца делаются следующие проводки (рис. 1.12):

- **62/90,**
- **90/41,**
- **90/68.**

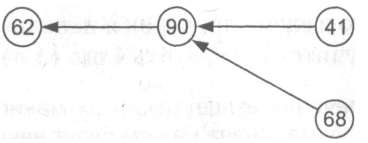


Рис. 1.12. Проводки по закрытию счета 90

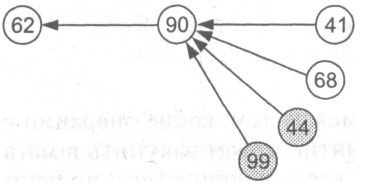


Рис. 1.13. Полная схема закрытия счета 90 с учетом проводок, формируемых в конце месяца

Кредитовая часть счета 90 — это отгрузки по продажным ценам. Дебетовая часть — это себестоимость товара и НДС отгруженного товара. Разница кредитового оборота счета и дебетового оборота счета равняется предполагаемой наценке.

В конце месяца добавляются проводки затрат и прибыли (рис. 1.13).

Проводка 90 / 99 закрывает счет 90. Закрытие заключается в выравнивании дебетовой и кредитовой частей счета 90. Если проводка получилась отрицательной, это значит, что предприятие сработало в убыток, а если проводка получилась положительной, то предприятие сработало с прибылью.

Издержки обращения

Вы уже встречались со схожим счетом в бухгалтерском учете (20, 26,...), однако двадцатые счета используются для списания затрат производства, а счет 44 («Издержки обращения») используется для сбора затрат торговли. В конце месяца счет 44 закрывается на счет 90.

Обычно счет 44 закрывается автоматически документом «Закрытие месяца».

Как превратить товары в материалы?

Некоторые производственные фирмы одновременно являются торговыми. Например, фирма может заниматься одновременно оптовой торговлей и производством кондитерских изделий. На таких предприятиях один и тот же товар может быть как товаром (счет 41), так и материалом (счет 10).

Чтобы превратить товары в материалы, следует сделать документ, который бы формировал проводки **10/41.**

Возможна еще одна трудность, которая заключается в том, что на счете 41 товары находятся в одних единицах, например в бутылках, а на счете 10 материалы учитываются в граммах. Для решения этой проблемы следует проектировать документ, который списывает товары в одном количестве и оприходует материалы в другом количестве, например так:

- **16/41/100-00/1шт.,**
- **10/16/100-00/1000 гр.**

Обратите внимание, что в схеме проводок я использую буферный счет 16. Без его использования не получится превратить 1 шт. (1 л) в количество 1000 гр.

ПРИМЕЧАНИЕ Для обеспечения вышеуказанных манипуляций счет 16 не должен иметь количественного учета, иначе на нем будут накапливаться остатки.

Заметим, кстати, что проводки 10 / 41, 16 / 41 и 10 /16 формируются на суммы без НДС.

Розница

Существует путаница между тем, какие операции считать розничными, а какие — оптовыми. Предприятие может закупить в магазине большое количество товара, расплатившись на кассе наличными и получив товарный чек, — и такая операция будет розничной. В то же время покупатель может выписать счет на оплату и оплатить счет через банк (или через приходный кассовый ордер), и эта операция будет оптовой. Оптовые и розничные операции различаются тем, что в опте ведется аналитический учет по покупателям, а в рознице покупатели остаются безличными. Схема проводок в розничной торговле приведена на рис. 1.14.

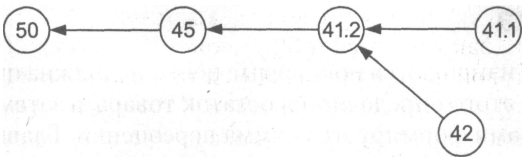


Рис. 1.14. Схема проводок в розничной торговле

Документооборот в оптовой торговле будет следующий:

- счет на оплату (может отсутствовать),
- расходная накладная,
- счет-фактура,
- доверенность на получение товара.

Оптовая операция может быть растянута во времени, то есть занимать более одного дня.

Документооборот в розничной торговле состоит из одного документа — товарного чека. Розничные операции, в отличие от оптовых, являются «мгновенными»: отдал деньги — получил товар. В розничном учете не может образовываться задолженность на счетах клиентов и ни одна из сторон не кредитует другую (то есть не «дает в долг» товар без оплаты или деньги без поставки товара).

Кстати, в бухгалтерском учете под словом «касса» понимается два совершенно различных объекта:

- «касса» в смысле «Кассовая книга», в которой фиксируются приходные и расходные кассовые ордера;
- «касса» в смысле места, где пробиваются чеки при совершении покупок, например при покупке хлеба.

Различия оптового и розничного учета сведены в табл. 1.3.

Таблица 1.3. Различия оптового и розничного учета

	Оптовый учет	**Розничный учет**
Количественный учет	Ведется	Может не вестись
Суммовой учет	Ведется по закупочным ценам (без НДС)	Ведется по продажным ценам с учетом НДС и наценки
Характер переоценки товаров	Переоценка номинальная, то есть производится замена одних ценников на другие	Формируется специальный документ, отражающий сумму переоценки
Отражение клиентов в документах	Ведется	Не ведется

Ведение розничного учета позволяет сократить документооборот предприятия, поскольку обеспечиваются следующие управленческие решения:

- В рознице не нужно отслеживать движение каждого товара.
- Розничный учет позволяет упростить контроль продавцов и свести его к сверке суммы остатка товарного отчета с суммой фактической по инвентаризации склада.

Переоценка

Когда в рознице изменяются продажные цены, то должна производиться переоценка товара. Для этого определяется остаток товара, а затем на разницу между старой и новой ценами формируется сумма переоценки. Главное в переоценке — определить сумму, на которую изменяется товарный остаток на складе.

Из того, что в розничном товарном учете весь товар учитывается в суммовом выражении, следуют следующие выводы:

- Если в розничной торговле производятся скидки, то вместе со скидкой следует делать и переоценку товара, что серьезно усложняет работу операторов, выписывающих накладные.
- Если в рознице находится один тип товара, но по разным ценам, то следует вести учет остатков товаров по всем типам цен.

Как удостовериться, что клиент говорит именно о рознице?

Когда клиент утверждает, что у него идет розничный учет цен, следует убедиться в этом, задав уточняющие вопросы из следующего списка:

- Правильно ли я понял, что вы учитываете товар по продажным ценам?
- Скажите, пожалуйста, когда вам надо продать товар со скидкой, вы каждый раз составляете акт переоценки?
- Не могли бы вы объяснить, почему вы не пользуетесь оптовой схемой учета цен?
- Если не сложно, расскажите, как составляются товарные отчеты продавцами и каким образом вы их в настоящее время обрабатываете.

Если оказывается, что переоценка товаров (смена цен) на самом деле является сменой ценников (или составлением нового прайс-листа), а специальный документ с внесением остатка и определением суммы переоценки не составляется, то или клиент понимает под розничным учетом что-то свое, или в его учете царит полный бардак.

ПРИМЕЧАНИЕ Розничная торговля хорошо реализована в конфигурации «Торговля» 9.*.

На самом деле учет в розничных ценах может быть и без наценки и НДС. Внедрению этой схемы способствует компьютеризация торгового учета. В этом случае учет в рознице ведется так же, как и в опте, через счет 90.

Если количественный учет в рознице не интересует

Вполне может оказаться, что владельца розничной сети товарный учет не интересует. В этом случае бухгалтерский учет сводится к учету задолженности поставщиков (счет 60) и укрупненному бухгалтерскому учету счета 41.2.

ПРИМЕЧАНИЕ Конфигурация, в которой реализуется розничный учет в суммовом выражении, находится в файле File006.zip в материалах к данной книге по адресу www.piter.com/download. В этой конфигурации следует смотреть документ «Товарные отчеты розницы».

Вмененный налог

С целью сокращения налоговых выплат и упрощения системы документооборота предприятия, торгующие в розницу, используют налоговый режим работы Единого налога на вмененный доход (ЕНВД). ЕНВД платится с площадей, на которых находится розничная точка.

Оптовая торговля

В оптовой торговле используется следующий цикл проводок (рис. 1.15):

- 41 / 60 — поступление товара (без НДС);
- 19 / 60 — поступление НДС;
- 90 / 41 — списание товара по себестоимости (без НДС);
- 90 / 68 — НДС отгруженный;
- 62 / 90 — выставление задолженности.

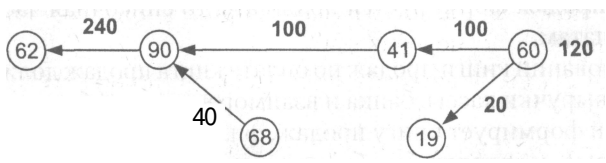


Рис. 1.15. Цикл проводок в оптовой торговле

Книга продаж

Книга продаж — это документ, который содержит информацию, подтверждающую суммы отгруженного НДС.

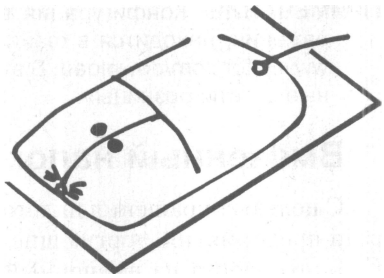
В стандартной конфигурации «Бухгалтерии» определена следующая цепочка документов при отгрузке товаров покупателю:

- расходная накладная (проводки 90 / 41),
- счет-фактура (проводки 90 / 68) — документ, по которому делается запись в книге продаж.

Схема отгрузки может быть несколько другой:

- расходная накладная (проводки 90 / 41),
- счет-фактура (проводки 90 / 76),
- запись книги продаж (проводки 76 / 68) — документ, по которому делается запись в книге продаж.

В книгу продаж попадают документы, Которые формируют проводки ** / 68. Если нужно сделать так, чтобы запись в книге продаж формировалась не по



счету-фактуре, а несколько позже, то используется вторая схема отгрузки. Например, такая ситуация происходит при оплате.

Возможно, у вас возник вопрос, нельзя ли расходную накладную и счет-фактуру объединить в один документ. Ответ кроется в правилах ведения бухгалтерского учета. Выданные счета-фактуры должны иметь сквозную нумерацию (то есть не должны иметь пропусков). Некоторые виды расходных накладных не влекут за собой формирование счетов-фактур. Так, розничная расходная накладная не формирует счета-фактуры.

Расходная накладная — возврат поставщику — влечет за собой запись не в книгу продаж, а запись в книгу покупок. На эту операцию должна формироваться отрицательная запись в книгу покупок.

Кстати, многие программисты, несмотря на вышесказанное, упрощают себе задачу и объединяют накладные и счета-фактуры в один документ.

Книга продаж может формироваться по отгрузке и по оплате.

Если книга продаж формируется по отгрузке, то книга продаж — это реестр счетов-фактур.

Если книга продаж формируется по оплате, то списочная часть должна соответствовать оплатам.

При формировании книги продаж по оплате книга продаж должна составляться из документов выручки кассы, банка и взаиморасчетов. Я бы советовал включить в отчет, который формирует книгу продаж, фильтры «Касса», «Банк» и «Взаиморасчеты», которые позволяли бы сбить книгу продаж с выручками соответствующих видов оплат.

ПРИМЕЧАНИЕ Отчет, который формирует книгу продаж по оплате без использования стандартных регистров, находится в файле File007.zip в материалах к данной книге по адресу www.piter.com/download.

Алгоритм книги продаж основывается на хронологическом закрытии документов (подробнее об этом алгоритме смотрите в разделе «Хронологическое закрытие накладных»).

Стоит сказать пару слов, почему не годилась книга продаж, предусмотренная в стандартных отчетах. Дело в том, что стандартная книга продаж выдавала данные, которые бухгалтер не мог проверить, а программист объяснить. Отчасти это происходило из-за того, что клиент правил документы задним числом. Мотивы правки были следующими:

- В случае оплаты безналичным расчетом следовало убирать налоги с продаж в накладных.
- В случае неправильно разнесенных оплат бухгалтеры принимали решение исправлять первичные документы оплаты.

Конечно, можно было бы после каждой правки восстанавливать последовательность документов. Однако восстановление последовательности создавало другую проблему. Размеры файлов обмена баз данных для работы с распределен-

ными базами данных достигали десятков мегабайт, что парализовало работу филиалов при поступлении обмена данных с восстановленной границей последовательности.

НДС

В Плане счетов НДС встречается дважды:

- счет 19 — «НДС по приобретенным ценностям»;
- счет 68 — «Налоги и сборы (НДС)».

Счет 19 отвечает за поступающий в предприятие НДС. Счет 68 отвечает за «отгружаемый» с предприятия НДС.

Схема использования этих счетов приведена на рис. Г.15. Поступивший на сумму 120 р. товар (100 р. — товар, 20 р. — НДС) был отгружен на сумму 240 р. (200 р. — товар, 40 р. — НДС). Причитающаяся сумма оплаты в бюджет будет определяться разницей между оборотами счетов 68 и 19 (40 - 20 = 20 р.).

Хитрости, связанные с НДС

Не всегда можно поставить НДС на счет 19. Например, когда товар закуплен в розницу, то НДС нельзя оприходовать. Банковские операции также не облагаются НДС. (Слова «не облагаются НДС» означают, что вы должны будете оплачивать НДС не банку, а государству при составлении налоговой декларации, а не то, что НДС вообще не должен оплачиваться.)

Хитрый пример 1

Представьте, что вы получили выручку от покупателей в размере 100 р. Банк снял с расчетного счета 100 р. за банковское обслуживание. Позже, при составлении налоговой декларации по НДС, вы должны будете заплатить еще 20 р. НДС. Интересно, правда?

Хитрый пример 2

Когда говорят, что минута разговора стоит 10 р. без НДС, это значит, что пользователь должен будет заплатить еще 2 р. НДС, когда речь зайдет о реальных оплатах, то есть стоимость минуты будет равна 12 р.

Государство разделило счета НДС полученного и НДС отгруженного, чтобы никогда не быть внакладе и чтобы предприятие всегда оплачивало причитающийся уплате НДС.

Внутри предприятия все материальные ценности хранятся без НДС, удивительно, правда? Здесь уместна такая модель. Следует провести интегрирующий контур вокруг предприятия (рис. 1.16). Все, что окажется внутри предприятия, учитывается без НДС, а все связи с внешним миром делаются с учетом НДС. Согласно этой схеме внутренние операции между складами должны производиться по закупочным ценам без НДС.

Хитрый пример 3

1. Предприятие купило товар *без НДС* за 120 р. у другого предприятия, которое работает по упрощенной схеме налогообложения.

2. Затем товар был продан *с НДС* за 122 р., причем само предприятие является плательщиком НДС.

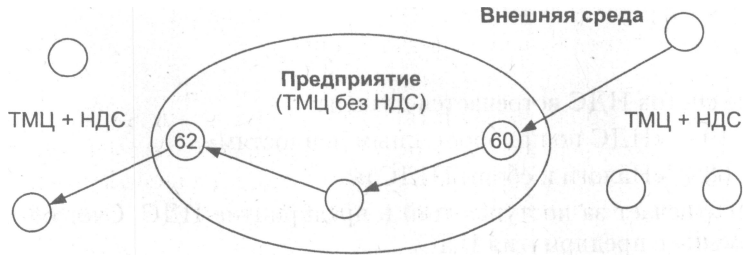


Рис. 1.16. Внутренние операции предприятия производятся без учета НДС, операции с внешними предприятиями (узлы вне овала) производятся с учетом НДС

Требуется определить доход, полученный в результате сделки. Если вы ответили, что прибыль будет 2 р. (122 - 120 = 2), то вы ошиблись, так как не учли НДС, которого нет при покупке, но есть при продаже. На самом деле в результате сделки будет получен убыток в размере 18 р.

Как считается наценка?

При ответе на вопрос, как считается наценка, напрашивается модель из мира потребителей, а именно: следует из продажной суммы вычесть закупочную сумму, и полученный в результате вычитания результат и будет являться наценкой. Так делают управленцы и ошибаются примерно на 20 %, то есть на сумму НДС. Правильно поступать по-другому — по-бухгалтерски. Для правильного расчета наценки следует из суммы реализации *А* (суммы, отгруженной покупателю) вычесть НДС отгруженный покупателю *С* и себестоимость поступившего товара *В:*

- оборот между счетами 62 / 90 (Реализация) — Л,
- оборот между счетами 41 / 60 (Себестоимость поступившего товара) — B_j
- оборот между счетами 90 / 68 (НДС отгруженный) — С,
- *Наценка* = *А* - *В* - *С*.

Так, если вспомнить пример на рис. 1.15, наценка составит следующую сумму:

Наценка = 240 *(А)* - 100 *(В)* - 40 (С) = 100 р.

Принцип рассуждений следующий: НДС, акцизы (и другие косвенные налоги) являются собственностью государства, а собственность государства не может относиться на доходную часть предприятия.

При составлении программ хорошо задавать себе вопрос: каким образом в схеме должен участвовать НДС? Например, производится передача товаров с оптового склада (счет 41.1) на склад материалов (счет 10). Будет ли в суммах учитываться НДС?

Ответ: в сумме перемещения суммы НДС участвовать не будет, поскольку во внутренних операциях НДС не учитывается.

Какие цены бывают?

Перечислю возможный, но далеко не исчерпывающий список возможных цен:

- цены с НДС и без НДС;
- закупочные, продажные и учетные цены;
- продажные со скидкой или с наценкой.

Следует быть осторожным в употреблении термина «цена». Если вы слышите слово «цена», будьте уверены, что это слово обозначает совсем не то, что вы думаете. Чтобы избежать путаницы, я бы советовал всегда добавлять уточняющее определение, например «закупочные цены», «отпускные цены», «цены себестоимости», «розничные цены».

Постарайтесь исключить из своего лексикона термин «продажные цены», поскольку этот термин включает все перечисленные типы цен.

Кстати, с точки зрения бухгалтера цен не бывает, можно говорить только о *количестве* и *сумме,* а цены — это производная часть от этих понятий (отношение суммы к количеству).

Если речь идет о ценах, то желательно уточнять, какой бухгалтерский счет имеется в виду. Если речь идет о счете 41.1 (оптовом), то мы говорим о закупочных ценах без НДС. Если речь идет об отгрузочных документах (счет 90), то мы будем иметь дело с ценами в расходной накладной.

Пример путаницы с ценами

Допустим, на счете 10 («Материалы») осталось 9 ручек на сумму 100 р. Цена одной ручки будет равна 100/9= 11 р. 11 коп. Если списывать ручки по этой цене, то после полного списания всех ручек со счета останется одна копейка:

*100р. (начальный остаток) - 9 (ручек) * 11,11 = 0,01,*

то есть конечный остаток равен 0 ручек на сумму 0,01 р.

Правильно поступать по-другому: в момент списания ручек определить, не списывается ли полный остаток товара, и если это так, то списывать всю сумму, находящуюся на счете; если же списывается не полный остаток, то сумму списания определять по формуле:

Сумма списания = Количество списания * (Остаток товара в рублях / Остаток товара в количестве).

Как быть с «несходилками» в ценах счетов-фактур?

Проблема проявляется в счетах-фактурах, когда продажные цены учитываются с учетом НДС.

Допустим, что продано 10 единиц товара на общую сумму 99,99 р. НДС (18 *%)* включен в эту сумму. Тогда в счете-фактуре будут отражаться следующие цифры:

- Количество — 10;
- Цена (тариф) за единицу измерения — 8,47 р.;
- Стоимость товаров (работ, услуг), всего без налога — 84,74 р.;

- Сумма налога — 15,25 р.;
- Стоимость товаров (работ, услуг), всего с учетом налога — 99,99 р.

Как можно заметить, цена единицы товара равна 8,47 р., а стоимость товара без налога (10 шт. по 8,47 р.) — 84,74 р., а не 84,7. Однако никакой ошибки нет, поскольку цена рассчитывается обратным счетом.

Попробуйте поменять в Конфигураторе точность поля Цены до четырех знаков после запятой, и ошибка все равно рано или поздно проявится.

Для того чтобы не возникало подобных проблем, следует формировать цены так, чтобы цена товара делилась на 18/(100 + 18) без остатка для НДС 18 % и на 1/11 (10/(100 + 10)) - для НДС 10 %.

Кстати, клиенты вполне могут ввести в заблуждение программиста утверждением, что до сих пор проблем с ценами у них не было вообще, а после его прихода все стало плохо. Дело в том, что проблемы с ценами проявляются только на больших суммах.

Пример решения проблемы цен

```
Функция КрасиваяЦена18Процентов (Цена)
рез = Цена;
Нач = окрЩена*(18/118),2)*(18/118);
Кон = (окрЩена*(18/118),2) + 0.01)*(18/118);
Если (Кон - Цена) > (Цена - Нач) Тогда
    рез = Нач;
Иначе
    рез = Кон;
КонецЕсли;
возврат рез;
КонецФункции
```

Акт сверки

Как сверить данные бухгалтерского учета с фактическим состоянием системы? Для сверки данных с поставщиками и покупателями составляются акты сверки, а для сверки состояния материальных ценностей проводится инвентаризация.

Инвентаризация приводит в соответствие учетное и реальное количество МЦ.

С течением времени происходит рассогласование фактического состояния системы с данными бухгалтерского учета (рис. 1.17). Рассогласование учетов может быть связано с ошибками учета, кражами или пересортицей. Инвентаризация уравновешивает эти два процесса. Урегулирование остатков осуществляется оприходованием излишков или списанием недостачи.

Как это ни странно, но иногда клиенты (имеется в виду клиенты программистов) утверждают, что можно обойтись без актов сверки, а все дело в работоспособности программы. Работники таких клиентов объясняют несоответствие реальных остатков и отраженных в базе данных тем, что программное обеспечение работает некорректно. На это можно заметить, что чем мутнее вода, тем легче в ней ловить рыбку, то есть кому-то выгоден бардак. Правильность работы компьютера проверяется очень просто: формируется компьютерная распечатка и сравнивается с реальными документами. Но это лишь один из участков, на котором возможны сбои. Так, сбои при отгрузке товаров могут быть на следующих уровнях:

Рис. 1.17. Инвентаризация приводит в соответствие учетное и реальное количество МЦ

- на уровне программы (написан некорректный алгоритм),
- на уровне оператора (выбит не тот товар),
- на уровне кладовщика (отпущен не тот товар),
- на уровне экспедитора (товар отпущен не тому клиенту),
- на уровне поставщика (пересортица была допущена поставщиком товаров).

Как видите, программист может повлиять только на один из пяти уровней.

Стоит заметить, что проведение актов сверки — это задача не программиста, а бухгалтерии или торгового отдела. Однако программист, работающий в области автоматизации бухгалтерского учета, — больше, чем программист. Если больше некому налаживать нормальную работу, то это должен сделать программист.

В качестве примера, подтверждающего необходимость регулярных актов сверки, приведу ситуацию, случившуюся у одного из клиентов. Директор обнаружил несоответствие между данными покупателя и теми, что он ведет для себя. В результате расследования было выяснено, что работник, принимавший деньги, мошенничал, подправляя документы. Чтобы исключить повторение обмана, директор принял решение каждый месяц проводить акты сверки. В случае, если агент не предоставлял акт по какому-либо из клиентов, он не получал зарплату. Работники поволновались-поволновались и через три месяца стали воспринимать ситуацию ежемесячных актов сверки как нормальную.

Инвентаризация остатков товаров

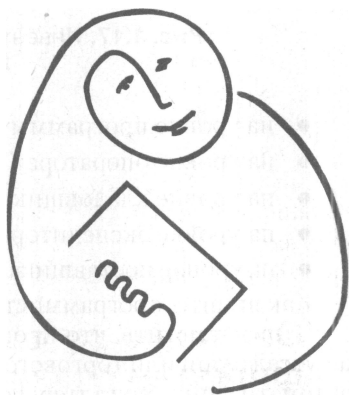
Когда требуется сделать инвентаризацию, необходимо выполнить две управленческие задачи: с одной стороны, следует начать «новую жизнь», оприходовав правильные остатки, а с другой стороны, обеспечить процесс урегулирования пересортицы и возможных ошибок учета.

Чтобы отделить «прошлую» жизнь от «будущей», достаточно ввести новый склад. Допустим, требуется провести инвентаризацию по группе «Косметика». Для этой цели можно создать склад «Склад новой косметики», на который следует оприходовать выявленное количество товара. «Прошлые» количества останутся на старом складе. Выписка новых документов теперь будет вестись с нового склада, а урегулирование остатков, определение пересортицы излишков и недостачи — на старом.

Такая схема проведения инвентаризации избавляет от риска, что урегулирование пересортицы в периоде до проведения инвентаризации разрушит остатки после инвентаризации.

Кроме этого, такая схема позволяет производить продажи сразу после проведения инвентаризации, не задаваясь вопросом, была ли выписана расходная накладная до инвентаризации или после нее.

Основные требования к ведению бухгалтерского учета



К ведению бухгалтерского учета предъявляются следующие требования:

- Ведение бухгалтерского учета осуществляется только на основании первичных учетных документов, оформляемых при проведении хозяйственных операций. Эти документы должны составляться по унифицированным стандартным формам, утвержденным Госкомстатом РФ (а при их отсутствии — разрабатываться самой организацией), и иметь ряд обязательных реквизитов.
- Учет должен вестись с использованием стандартного Плана счетов (на его основе организация может сформировать свой рабочий План счетов).
- Учет должен осуществлять проверку и документальное подтверждение данных бухгалтерского учета и отчетности путем проведения обязательной инвентаризации.
- Учет должен обеспечить учет имущества, обязательств и хозяйственных операций организации.
- Учет должен обеспечить соответствие данных аналитического учета оборотам и остаткам по счетам синтетического учета.
- Бухгалтерский учет должен вестись в рублях. Записи по валютным счетам и операциям в иностранной валюте должны производиться в рублях с пересчетом по курсу ЦБ РФ на дату совершения операции. Одновременно эти записи должны проводиться в валюте расчетов и платежей.

Как учитывать «черное» и «белое»?

Для многих является секретом, что от 30 до 70 % розничного или оптового оборота представляют собой различные формы черного оборота. Та же самая картина происходит на промышленных предприятиях. «Зачем платить несправедливые налоги?» — так рассуждают предприниматели и делают бухгалтерские балансы «Потемкинских деревень».

Существуют два варианта сокрытия доходов:

- сокрытие реализации;
- завышение затрат.

Сокрытие реализации актуально для мелких предпринимателей, которые являются бухгалтерами и предпринимателями в одном лице. У них все под контролем. Весь документооборот в голове или тетрадях. Закончил сделку и выкинул бумаги.

Крупным организациям трудно работать по такой схеме. Нужно одновременно и контролировать работников (вести бухгалтерский учет), и документооборот снижать. Поэтому реализуется другая схема — используются счета подставных фирм для того, чтобы обналичивать нужные обороты. Впрочем, существует и вполне легальная схема минимизации налоговых выплат, которая заключается в организации магазина, которого на самом деле нет.

Создается фирма, единственным имуществом которой является кассовый аппарат и стол бухгалтера. Такая фирма регистрируется как плательщик вмененного налога.

Вмененный налог рассчитывается с площадей, на которых находится магазин, — в нашем случае стол бухгалтера. Все «черные» продажи должны проходить через эту фирму. Правильнее сказать, что после того, как «обороты» пройдут через розничную фирму, они побелеют для продавца и останутся черными для покупателя.

Описание работы схемы «оптовая фирма — розничная фирма»

Организуются две фирмы: «Опт» и «Розница». В течение дня обеими фирмами выписываются документы с общего остатка.

На рис. 1.18 документы фирмы «Опт» показаны белыми прямоугольниками, а документы фирмы «Розница» — серыми прямоугольниками.

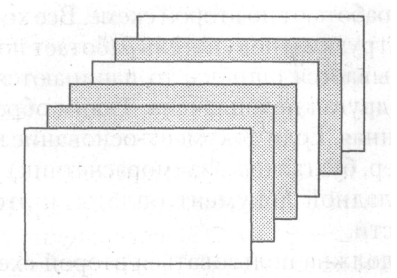


Рис. 1.18. В течение дня документы фирм выписываются с общего товарного остатка

В конце дня, для того чтобы фирма «Розница» не продавала воздух, а у фирмы «Опт» не возникло излишка товара, делается урегулирование остатков: формируется расходная накладная с фирмы «Опт» и приходная накладная на фирму «Розница» на количество продаж фирмой «Розница» (рис. 1.19).

К схеме остается добавить, что расходная накладная может формироваться с минимальной наценкой, чтобы весь доход оставался на розничной фирме.

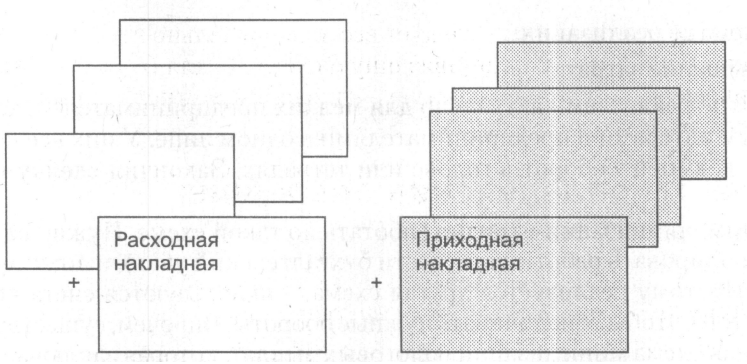


Рис. 1.19. В конце дня документы сортируются по признаку принадлежности к фирме. Урегулирование остатков производится дополнительными расходными и приходными накладными

ПРИМЕЧАНИЕ Обработка, анализирующая продажи с оптовой фирмы и формирующая расходную и приходную накладные, находится в файле File008.zip в материалах к данной книге по адресу www.piter.com/download.

Закрытие документов отгрузки

Незакрытые накладные можно определить двумя способами. Во-первых, незакрытыми следует считать последние отгруженные накладные *(хронологическая схема)*. Этот способ основывается на том, что клиент всегда оплачивает накладные в хронологическом порядке. Во-вторых, незакрытыми документами следует считать такие накладные, которые еще не привязаны к оплатам *(бухгалтерская схема)*. Этот способ основывается на том, что клиент оплачивает конкретный документ.

Многие предприятия работают по второй схеме. Все хорошо, когда суммы оплат эквивалентны суммам отгрузки и покупатель работает по этой же схеме. Когда же в процесс оплаты вкрадывается ошибка, то начинаются рассогласования: одна накладная переплачена, другая недоплачена. Таким образом, бухгалтерская схема менее помехозащищенная. Если документ-основание не указан, оператор (кассир, кредитный контролер, бухгалтер-взаиморасчетчик) на свое усмотрение привязывает к каждой накладной документ оплаты, и это становится причиной будущей неопределенности.

Однако бухгалтерия должна пользоваться второй схемой закрытия документов, потому что документы следует закрывать не так, как вздумается программисту или программе, а так, как написано в платежных документах. Но, с другой стороны, отчеты, составленные в правильной бухгалтерской форме, невозможно анализировать. И кроме того, не слишком крупные покупатели не всегда могут содержать высококвалифицированного бухгалтера, поэтому бухгалтер часто не следит за правильным указанием документа в платежном поручении.

Если вести учет по предписанной бухгалтерским учетом схеме, то отчет, который показывает задолженность клиента в разрезе незакрытых расходных накладных, может выдать не совсем наглядную картину. По одним накладным будет переплата, а по другим — излишний долг. Анализировать такой отчет невозможно.

Проблема решается просто: следует вести параллельно две системы закрытия накладных. Одну, правильно привязанную схему, — для бухгалтерии, а другую, хронологическую, лаконично простую, — для сиюминутного принятия решений, для работников торгового отдела.

Алгоритм определения последних незакрытых накладных

Определить последние хронологически незакрытые накладные гораздо проще, чем вы могли бы подумать. Для этого следует выбрать в обратной последовательности накладные на сумму текущей задолженности.

1. Определяем сумму текущей задолженности клиента (например, 100 р.).
2. Составляем список расходных накладных в обратном порядке. Первым в списке идет последний отгруженный документ, затем — предпоследний и т. д., например:

   01.12.03, Док.Ш 100,30-00,
   24.11.03, Док.ШОбО, 50-00,
   11.11.03, Док.т020,10-00,
   01.10.03, Док.М1005,40-00,
   01.09.03, Док.1ч[1001, 80-00.
3. Выбираем из списка документы на сумму задолженности:

   01.12.03, Док.Ш 100, 30-00,
   24.11.03, Док.ШОбО, 50-00,
   11.11.03, Док.Ш020,10-00,
   01.10.03, Док.Ш005,40-00.
4. Для определения просроченной задолженности следует отобрать из полученного списка все накладные, выписанные раньше определенной даты. Например, если следует выбрать все неоплаченные документы, начиная с 15.11.03, то получится следующий список документов:

   11.11.03, Док.Ш020,10-00,
   01.10.03, Док.Ш005,40-00.

После того как вышеуказанный алгоритм был внедрен в агентских структурах моих клиентов, проблема задолженности агентов для меня перестала существовать. До внедрения алгоритма каждый раз при смене состава агентов приходилось объяснять, что переплаты по накладным — это не ошибка программиста, а результат «правильной разноски оплат» бухгалтеров.

ПРИМЕЧАНИЕ Отчет, построенный по вышеуказанной схеме, находится в файле File009.zip в материалах к данной книге по адресу www.piter.com/download.

Хронологическое закрытие накладных

Одна из управленческих задач торгового учета состоит в определении срока оплаты накладных. На основании данных о сроках оплаты принимается решение об уровне вознаграждения торговых агентов или уровне скидок клиентам.

Для решения этой проблемы обычно используют стандартные регистры взаиморасчетов. Если отчет, основанный на использовании этого регистра, дает

некорректную информацию, то следует восставить границу последовательности документов. Хорошо, если база данных небольшая. Однако для большого предприятия восстановление последовательности оформления документов влечет за собой многочасовую остановку предприятия. Перепроведение документов также приводит к формированию больших по размерам файлов обменов с периферийными базами, и часто после перепроведения остатки по регистрам (например, отчеты по партиям) не совпадают с данными до перепроведения.

Как составить таблицу оплат в хронологическом порядке без использования регистров?

Следует дать пояснение, что эта таблица составляется для тех случаев, когда предприятие может менять задним числом накладные, но не может позволить себе восстановление последовательности оформления документов. Если предприятие работает правильно с точки зрения разработчиков информационных баз, то для решения данной задачи можно воспользоваться стандартным регистром задолженности.

Таблица составляется в соответствии со следующим алгоритмом:

1. Составляется список расходных накладных;
2. Составляется список оплат.
3. Строится таблица соответствия списка оплат списку отгрузок так, чтобы сумма отгрузки равнялась сумме оплаты.

ПРИМЕЧАНИЕ Отчет, построенный по вышеуказанной схеме, находится в файле File010.zip в материалах к данной книге по адресу www.piter.com/download.

Программист должен говорить на языке бухгалтера

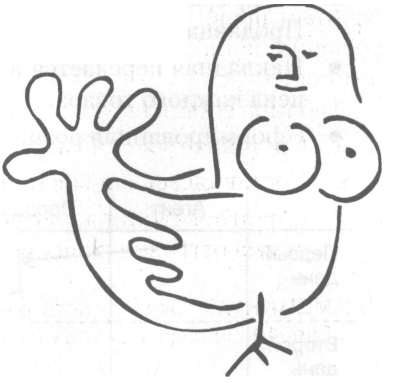
В данной главе фрагментарно описана теория бухгалтерского учета для программиста, который специализируется в торговом и производственном бухгалтерском учете. Для продолжения самообразования я рекомендовал бы прочитать несколько книг по бухгалтерскому учету. Пусть вы не поймете большую часть того, что там написано, но на интуитивном уровне вы научитесь говорить на языке бухгалтерии и сможете задавать клиентам такие вопросы, как будто для вас бухгалтерия — родная стихия:

- А в акте сверки обороты по каким счетам требуется учитывать?
- А за какие счета вы отвечаете?
- Ваше предприятие работает по оплате или по отгрузке? (Это сокращенная форма от вопроса: «Ваше предприятие платит НДС по отгрузке или по оплате?»)
- А вы закрываете счет 90 вручную?

Чем больше ваш язык приближен к языку специалиста, с которым вы общаетесь, тем меньший психологический барьер будет вас разделять.

Глава 2
Примеры постановок технических заданий

Две фирмы в бухгалтерском учете



Под *многофирменным учетом* я буду понимать систему, которая в одной базе данных позволяет вести операции для разных фирм (разных хозяйственных субъектов).

В бухгалтерском учете многофирменный учет в одной базе данных может быть организован с использованием разделителя учета. Для этого следует выполнить следующие действия:

1. Создать справочник Фирмы.
2. В общие реквизиты документов ввести поле ФРМ (тип Справочник Фирмы).
3. В объекте Проводка ввести новый реквизит Фирма (тип Справочник Фирмы).
4. В форме объекта Проводка щелкнуть на кнопке Формы журнала, выбрать ДляОпераций и внести реквизит Фирма.
5. В свойствах Плана счетов в поле Разделитель учета выбрать тип разделителя учета Фирма.

Подготовительный этап закончен. Теперь следует во все документы вставить реквизит Фирма и изменить программы формирования проводок. Для этого надо добавить в модуль проведения документа строку:

```
Операция.Фирма = ФРМ;
```

После этого следует перепровести документы и пересчитать итоги.

Пример многофирменного учета в «Бухгалтерии»

ЧП «Y» владеет сетью торговых центров. С целью минимизации расходов торговый центр «раздает» торговые площади родственным фирмам, то есть для проверяющих органов работает схема *«арендодатель* (юридическое лицо) — *предприниматели* (физические лица)», а на самом деле арендодатель и предприниматели представляют собой единую фирму с общим руководством, учредителями и бухгалтерией.

ПРИМЕЧАНИЕ Конфигурация, в которой реализован многофирменный учет при помощи компоненты «Бухгалтерия», находится в файле File011.zip в материалах к данной книге по адресу www.piter.com/download.

Многофирменный учет в «Торговле»

Многофирменный учет в конфигурациях, построенных с использованием компоненты «Торговля», заложен с самого начала. Разделение между фирмами уже реализовано, поэтому программисту не нужно ничего доделывать.

Передача документов по почте

Обычная схема работы торговой точки с широким ассортиментом товара следующая (рис. 2.1):

- Агент оптовой фирмы приезжает на торговую точку (Продавец) и отгружает товар (передает товар и расходную накладную).
- Товар сразу продавать нельзя, поскольку бухгалтерия торговой точки (офис Продавца) должен определить розничные цены.
- Накладная передается в офис Продавца. В офисе формируется розничная цена каждого товара.
- Сформированная розничная накладная поступает на розничную точку.

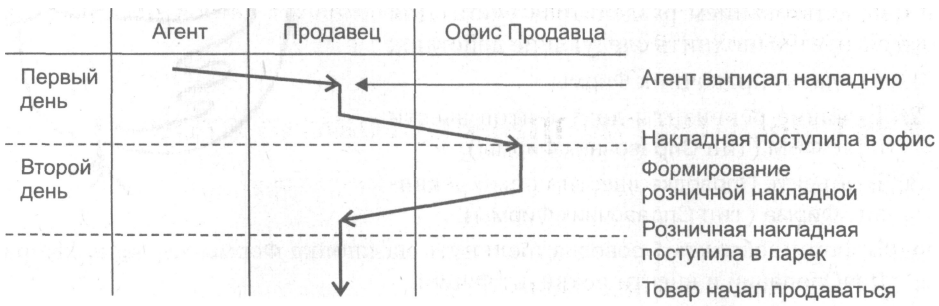


Рис. 2.1. Схема работы торговой точки

Возникает вопрос: как можно усовершенствовать схему, чтобы на операционные расходы уходило не полтора дня, как обычно, а день? Задача была успешно реализована при помощи внедрения интернет-технологий.

Сокращение операционных расходов может быть основано на принципе: «Один документ должен вноситься в компьютер один раз». Дело в том, что агент составляет накладную в электронном виде, а продавцу выдает ее бумажную копию. В новой системе документооборот изменяет схему (рис. 2.2):

- Документы из компьютера Агента поступают в информационную базу Продавца вечером, когда Агент возвращается в офис.
- Документы Продавцы сохраняются в файле и отправляются по электронной почте в офис Продавца.
- Документы принимаются из Интернета на компьютере в офисе Продавца.
- На товары формируется торговая наценка. На поступивший товар формируются ценники.

Рис. 2.2. Усовершенствованная схема работы торговой точки

После внедрения системы получилась экономия времени на обработку документов в полдня. Бухгалтер, ответственный за обработку документов в розничной сети, стал заниматься разноской документов не целый день, как раньше, а только один час в начале рабочего дня и при этом перестал ошибаться при вводе сумм, наценки на товар и при выборе товара.

ПРИМЕЧАНИЕ Конфигурация, демонстрирующая загрузку файла накладной в программу, находится в файле FileO12.zip в материалах к данной книге по адресу www.piter.com/download.

Учет работы группы программистов

Обычно у одной группы программистов есть определенный круг клиентов, которых они обслуживают совместно. В связи с этим возникают управленческие задачи, которые должны дать ответ на следующие вопросы:

- Какие задачи решались у клиента?
- Насколько срочными были эти задачи?
- Сколько времени каждый программист провел у клиента?
- Сколько всего каждый программист отработал времени?
- Какие задачи и сколько времени решал программист?

Учет должен вестись в количестве потраченного времени и в суммовом выражении (сколько клиент заплатил за услуги).

Все перечисленные задачи можно решить введением забалансового счета (счет, который участвует в проводке без счета-корреспондента). К счету необходимо присоединить следующие виды субконто: *клиент, сотрудник* и *вид работы*. Количественный учет будет соответствовать количеству потраченного времени, а сумма — количеству полученных от клиента денег.

Использование забалансового счета объясняется тем, что счет не находится в балансе, то есть увеличение этого счета не влечет за собой уменьшения каких-либо других счетов.

Введение вышеуказанной схемы бухгалтерских счетов и конструирование документа, облегчающего работу заполнения дневного отчета, займет не более 15 минут. Кстати, систему отчетов проектировать не надо: подойдут стандартные

отчеты, предусмотренные стандартной конфигурацией бухгалтерского учета (оборотно-сальдовая ведомость по счету, анализ субконто, карточка субконто).

ПРИМЕЧАНИЕ Конфигурация, реализующая описанную выше схему, находится в файле FileO13.zip в материалах к данной книге по адресу www.piter.com/download.

Постановка задачи для предприятия «X»

Каждая схема проводок описывается двумя формами представления: в виде таблицы проводок и в виде графа (схемы). Форма представления в виде графа позволяет наглядно определить, в каких пропорциях должны списываться средства.

Согласно требованиям главного бухгалтера компании «X» был существенно изменен План счетов. Так, для взаиморасчетов по предоплате был задействован счет 45.

Предоплатная схема проводок

Под *предоплатной схемой* принимается случай, когда покупатель сделал оплату (например, по счету) и по прошествии некоторого времени получил товар.

Расходная накладная

Кр	Дб
45	41
45	98

Оплата

Кр	Дб
51, 50	90
90	45
98	45
90	68
90	99

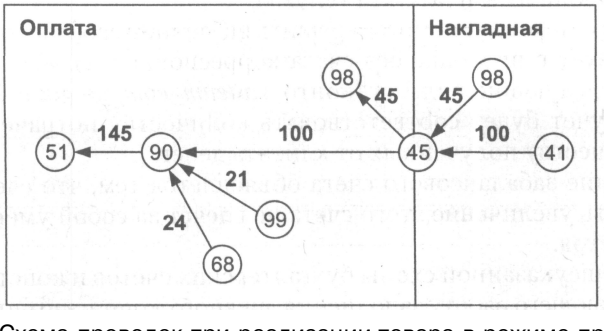


Рис. 2.3. Схема проводок при реализации товара в режиме предоплаты

Послеоплата

Под *послеоплатной схемой* понимается случай, когда предприятие поставляет клиенту товар, а клиент через некоторое время оплачивает поставку.

Отгрузка (проводки расходной накладной)

Кр	Дб
62	90
90	41
90	99
90	68

Оплата

Кр	Дб
51, 50	62

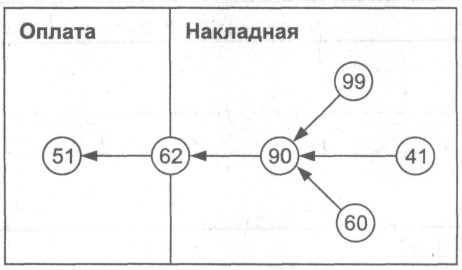


Рис. 2.4. Схема проводок при реализации товара в режиме послеоплаты

Розница

Розница — реализация товаров с использованием варианта реализации товаров через Вмененный налог.

Отгрузка (расходная накладная) в розницу

Кр	Дб
41.2	41.1
41.2	99
41.2	19

Оплата розницы

Кр	Дб
90	41.2
50	90

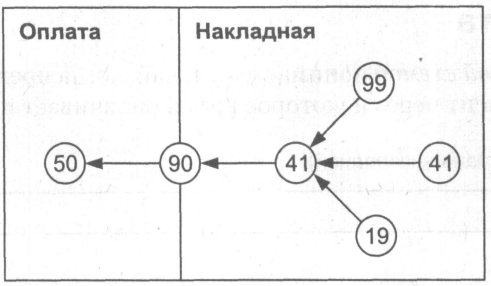


Рис. 2.5. Схема проводок при реализации товара в розницу

Пересортица во взаиморасчетах

Если требуется погасить долг на счете 62 или 60 за счет предоплаты (счет 45), то должны формироваться такие же проводки, как и в предоплатной схеме (при поступлении денег в кассу или банк), только вместо счета 50 или 51 следует ставить счет 62 или 60:

Кр	Дб
62, 60	90
90	45
98	45
90	68
90	99

ПРИМЕЧАНИЕ Формировать проводку 62/45 нельзя, поскольку нужно сторнировать неправильно сформированный в этом случае НДС.

Книга продаж

Книга продаж должна отражать следующие проводки:

Кр	Дб
90	68

Проводка 90/68 формируется отгрузочной накладной в случае послеоплаты и документом оплаты в случае предоплаты.

Книга покупок

Книга покупок для этой постановки задачи — это реестр приходных накладных. Возможно, у вас возникнет вопрос, почему книгу продаж нельзя было бы

сделать точно так же, как книгу покупок — реестром. Ответ заключается в количестве видов документов, необходимых для обработки.

В книгу продаж попадают как минимум три вида документов:

- оплата кассовая,
- оплата банковская,
- взаиморасчеты (ручные проводки).

В книгу покупок попадает один вид документа — приходная накладная.

ПРИМЕЧАНИЕ Конфигурация, в которой запрограммирована вышеприведенная схема, находится в файле FileQ14.zip в материалах к данной книге по адресу www.piter.com/download. Следует смотреть модуль проведения документов расходной и приходной накладной, а также приходного кассового ордера и банковской выписки.

Товарные отчеты розничной торговли «К»

Задача, поставленная руководством фирмы — владельца сети магазинов, была следующей: требуется неизменным количеством бухгалтеров обрабатывать в два раза больше магазинов. Проанализировав работу бухгалтеров, выяснили, что она ведется по следующей схеме:

1. Прежде всего, бухгалтер-взаиморасчетчик определяет документы, которые должны быть оплачены в тот же день после поступления товарного отчета из магазина! После этого кипа документов передается на второй этап, а бухгалтер-взаиморасчетчик определяет документы за прошлые дни, срок оплаты которых уже подошел.
2. Каждая строка товарного отчета расписывается на следующие составляющие:
   - o сумма товара без наценки,
   - o сумма торговой наценки,
   - o сумма тары,
   - o сумма услуг (например, сертификатов).
3. Согласно обработанному товарному отчету заполняются ручные проводки в «Бухгалтерию». В систему вносились только суммы (количественный и номенклатурный учет бухгалтерию не интересовал).

Для оптимизации работы «Бухгалтерии» было принято решение создать новый документ «Розница», который был бы максимально приближен к виду тех документов, которые бухгалтеры обрабатывают в бумажном виде.

Внедрение документа снизило трудоемкость работы бухгалтеров, что позволило тем же составом бухгалтерии обрабатывать большее количество вновь появившихся магазинов.

ПРИМЕЧАНИЕ Конфигурация, реализующая приведенную выше схему, находится в файле FileO15.zip в материалах к данной книге по адресу www.piter.com/download. В конфигурации следует смотреть документ «Розница».

Учет возвратной тары

Учет возвратной тары в оперативном (торговом) учете

Для ведения учета возвратной тары был введен регистр «Кеги», а к нему присвоили следующие измерения:

- Товар,
- Клиент.

ПРИМЕЧАНИЕ Конфигурация, в которой реализован учет кег, находится в файле FileO16.zip в материалах к данной книге по адресу www.piter.com/download. Следует обратить внимание на следующие объекты конфигурации: регистр «Кеги», модуль документов «Расходная накладная» и «Приходная накладная», процедура глобального модуля «Кеги», документ «ИнвентаризацияКег», отчет «Кеги».

Учет возвратной тары в «Бухгалтерии»

Учет возвратной тары в бухгалтерском учете организовать несколько проще, чем в торговом учете. Для этого следует организовать забалансовый счет, присвоить одному субконто счета тип «Номенклатура», другому — тип «Клиенты», а также дописать процедуру формирования проводок в расходную и приходную накладные. Инициализацию данных на счете можно выполнить вручную, а для анализа использовать стандартные отчеты.

ПРИМЕЧАНИЕ Структура проводок и счета будет аналогична той, что приведена в разделе «Учет работы группы программистов».

Пакетная печать документов

Обычно расходные накладные, счета-фактуры и кассовые документы печатаются на лазерном принтере. Себестоимость одного листа, напечатанного на лазерном принтере, не превышает 50 копеек.

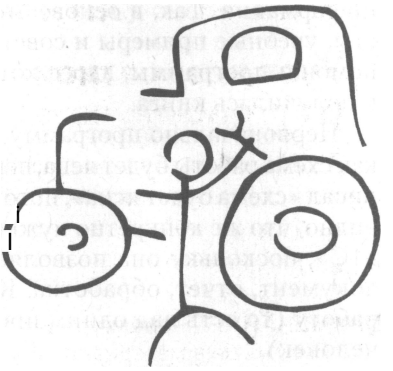
Стоимость подготовки одного документа можно сократить, если использовать матричный, а не лазерный принтер. Матричные принтеры менее требовательны к качеству бумаги, а расходные материалы для матричного принтера ничтожно малы по сравнению с расходными материалами для лазерного. Одна беда: матричные принтеры печатают долго и к тому же довольно шумно.

Возникает творческая задача: как сократить время распечатки, избавиться от шума и в то же время сэкономить деньги?

Решение 1. Заменять прямые линии псевдографическими последовательностями из знаков «-», «+» и «|». Тогда матричный принтер будет печатать строки за один проход, а не за два, как он это делает при печати прямых линий.

Решение 2. Упростить форму документов. Например, включать в документы, составляемые для склада, только ту информацию, которая необходима кладовщику (данные о клиенте, товаре и его количестве), и не печатать информацию об НДС и суммах.

Решение 3. Печатать не днем, а ночью, когда шум принтера никому не будет слышен.



Для реализации перечисленных выше инноваций был разработан отчет «Пакетная печать».

ПРИМЕЧАНИЕ Обработка, реализующая описанную выше схему, находится в файле FileO17.zip в материалах к данной книге по адресу www.piter.com/download.

Оператор вечером набирал документы и отправлял их на печать. Когда утром приходили работники офиса, их ждали готовые распечатки. Если же происходил сбой в работе принтера или требовались правки на складе, то исправленные документы печатались на лазерном принтере.

Переброска документов при помощи OLE

Программа, разработанная для учета торговых операций, может работать на протяжении пяти лет, в то же время бухгалтерские конфигурации менее долговечны. Изменяется законодательство, вводятся налоги с продаж и налоговый учет, меняются главные бухгалтеры. Каждый новый главный бухгалтер подвергает сомнению созданную предыдущим бухгалтером систему и желает начать учет с нового листа и с новой бухгалтерии.

Как быть? С одной стороны, требуется оставить конфигурацию торгового отдела прежней, а с другой стороны, конфигурацию бухгалтера требуется обнулять с приходом нового главного бухгалтера.

Для того чтобы решить эту проблему, можно разделить торговую и бухгалтерскую системы, а документы между базами данных переносить при помощи OLE.

ПРИМЕЧАНИЕ Реализация переноса данных при помощи OLE удобна тем, что если меняется структура данных переноса, то можно отлаживать одну программу, а не две (программу выгрузки данных в текстовый файл и программу загрузки данных).

Опыт показывает, что если переносятся данные из «Торговли» в «Бухгалтерию», то обрабатывающую программу лучше делать на базе данных «Бухгалтерии».

ПРИМЕЧАНИЕ Обработка, которая переносит данные (справочники Номенклатура и Контрагенты, расходные и приходные накладные, приходные кассовые ордера и банковские выписки), находится в файле FileO18.zip в материалах к данной книге по адресу www.piter.com/download.

Помощник писателя

Любая научная книга начинается с картотеки. В качестве содержимого карточек будут интересные мысли, примеры, ссылки на статьи и где-то услышанная

информация. Так, в основе этой книги первоначально были разрозненные тексты, учебные примеры и советы. Позже карточки были систематизированы при помощи программы, адрес которой приведен в конце статьи, в результате чего и получилась книга.

Первоначально программу было решено смоделировать в «1С», а после того как схема работы будет ясна, перевести ее на другой язык программирования. Я написал «схема будет ясна», потому что, когда появляется новая идея, не всегда очевидно, что же конкретно нужно делать. Для моделирования была выбрана среда «1С», поскольку она позволяет использовать стандартные блоки: справочник, документ, отчет, обработка. Кроме того, «1С» поддерживает распределенную работу (то есть над одним проектом может одновременно работать несколько человек).

Постановка задачи

1. Организовать справочник категорий.
2. Импортировать текстовые файлы в документы (организация картотеки предполагалась в виде документов).
3. К каждому тексту привязать некоторое количество категорий, которые характеризовали бы текст с разных точек зрения.
4. Провести систематизацию текстов и сгруппировать их по выявленным категориям.
5. Вывести в текстовый файл результаты работы.

Реализация задачи

В ходе ручной привязки категорий выявились трудности: слишком много времени уходило на то, чтобы для каждого текста (документа) выбрать соответствующий элемент из справочника категорий. Поэтому пришлось изменить постановку задачи. Ключевые слова я решил не выбирать из справочника, а набирать на клавиатуре. Например, анализируя текст про обучение, можно было напечатать такую строку ключевых слов: «обучение, харизма, личный рост».

Теперь написание ключевых слов стало занимать менее тридцати секунд. После того как ключевые слова для всех текстов были введены, была составлена программа — анализатор ключевых слов. Программа разбивала строку с ключевыми словами на отдельные элементы, которые затем заносила в справочник.

В результате моделирования появился прототип электронного помощника писателя. Главное в прототипе проекта — это скорость. Важно отработать основные идеи системы и сделать разработку приложения максимально быстрой. После того как прототип был готов, можно было писать техническое задание на проектирование программы и делать программу более приспособленными для этого средствами.

Позже идея «помощника писателя» была полностью пересмотрена и преобразована в программу TextIndexer. Смотрите описание программы здесь: www.citycat.ru/iq/ti. Там же находится ее бесплатная версия.

ПРИМЕЧАНИЕ Конфигурация «Помощник писателя» находится в файле FileO19.zip в материалах к данной книге по адресу www.piter.com/download.

Торговый проект

Клиент описал задачу следующим образом: «Покупателю выставляется счет на оплату, но при этом товара на складе еще нет. Я звоню поставщику товара, и он мне высылает свой счет. Позже поставщик выписывает дополнительные счета на оплату транспортных расходов. Следует составить программу так, чтобы я знал прибыль, возникающую в ходе сделки».

Вы что-нибудь поняли? Лично я — нет. Для уточнения Задачи был нарисован график взаимодействия фирмы, поставщика и покупателя (рис. 2.6). Рисунок показал, что различные документы следует относить к одному проекту и составлять двойной акт сверки (одновременно для покупателя и поставщика).

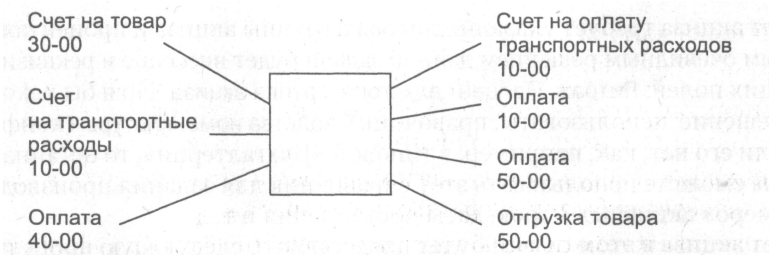


Рис. 2.6. Схема взаимодействия фирмы, поставщика и покупателя

Схема, поясняющая работу проекта, приведена в табл. 2.1.

Таблица 2.1. Торговый проект

Взаимоотношения с поставщиком		**Взаимоотношения с покупателем**	
Отгрузки поставщика	**Оплаты поставщику**	**Отгрузки покупателю**	**Оплаты покупателя**
30-00 (товар) 15-00 (транспорт)	30-00 (товар) 15-00(транспорт)	50-00 (товар)	50-00 (товар)
Проект закрыт			
Доход от сделки составил 5-00			

Для того чтобы реализовать эту схему, достаточно создать справочник Проекты и внести в поле накладных и документов оплаты поле с типом Справочник Проекты.

ПРИМЕЧАНИЕ В качестве основы решения проблемы можно воспользоваться схемой, которая реализована в торговых документах стандартных конфигураций «Торговли» 9.** (справочник Проекты).

Таблица «Клиент - Товар»

Иногда возникает необходимость в отчете, который имел бы циклы по строкам и столбцам. Ниже приведена ссылка на отчет, который печатает результат

в форме «Товар - Клиент». Список строк и столбцов заранее не определен. Пример демонстрирует использование запросов. Вероятно, вы будете удивлены, что такой отчет можно написать из 90 строк.

ПРИМЕЧАНИЕ Отчет, реализующий схему «Товар - Клиент», находится в файле File020.zip в материалах к данной книге по адресу www.piter.com/download.

Обратите внимание на следующие строки отчета:

```
(Группировка Тов без групп все ВошедшиеВЗапрос;
Запрос.ВначалоВыборки();
```

Расчет акциза

Расчет акциза требует указания литража группы акциза и процентажа алкоголя. Самым очевидным решением данной задачи будет внесение в реквизиты товара следующих полей: Литраж, Процент алкоголя, Группа акциза. Но я бы рекомендовал другое решение: использовать справочник Свойства номенклатуры (конфигурация 9.*), а если его нет, как, например, в типовой «Бухгалтерии», то организовать его. Позже вы сможете использовать этот справочник для задания производителя товара, номеров акцизных марок, даты поступления и т. д.

Расчет акциза в этом случае будет представлять следующую процедуру, которую можно разместить в глобальном модуле:

```
Функция Акциз(Тов, ДатаА) Экспорт
    сво = СоздатьОбъект("Справочник.СвойстваНоменклатуры");
    сво.ИспользоватьВладельца(Тов);
    сво.ВыбратьЭлементы();
    Литраж = 0;
    Алкоголь = 0;
    ГруппаАкциза = "";
    Пока сво.ПолучитьЭлемент() = 1 Цикл
        сво_ = сокрлп(сво.Свойство);
        Если сво_ = "ГруппаАкциза" тогда
            ГруппаАкциза = (Сво.Значение);
        ИначеЕсли сво_ = "Литраж" тогда
            Литраж = число (строка(Сво.Значение));
        ИначеЕсли сво_ = "Процент алкоголя" тогда
            Алкоголь = число(строка(Сво.Значение));
        КонецЕсли;
    КонецЦикла;
    Гру = СоздатьОбъект("Справочник.ГруппыАкциза");
    Гру.ИспользоватьДату(ДатаА);
    Гру.ВыбратьЭлементы();
    ст = 0;
    Пока Гру.ПолучитьЭлемент() = 1 Цикл
        Если Гру.ГруппаАкциза = ГруппаАкциза Тогда
            прервать;
        КонецЕсли;
    КонецЦикла;
    Акциз = (Алкоголь/100)*Литраж*Гру.Ставка1;
    Возврат Акциз;
КонецФункции
```

Как изменять конфигурацию, не изменяя ее?

Обычная трудность, с которой сталкивается программист в крупных организациях, заключается в том, что монопольный доступ к базе данных для обновления конфигурации можно получить или в конце дня, или рано утром.

Однако вы можете на ходу исправлять конфигурацию, если будете использовать «специальную конструкцию», которая указывает, из какого файла следует брать программный модуль:

```
# ЗагрузитьИзФайла РасходнаяНакладная.txt
```

Считывание файла происходит в момент открытия окна формы. Для представления внешнего файла в более привычной для редактирования форме используйте следующие команды: Файл • Открыть, Действия • Текст модуля.

Акты сверки

Акт сверки служит для приведения бухгалтерского учета в соответствие с фактическим состоянием и выявления возможных ошибок и краж.

В акте сверки следует предусматривать работу с двумя базами данных. Если база данных обрезалась, то следует сделать так, чтобы акт сверки компилировался из текущей и прошлой баз данных.

Акт сверки должен предусматривать режим объединения нескольких клиентов в одну группу. Часто бывает так, что несколькими клиентами владеет одно юридическое лицо. В этом случае следует предусматривать как отдельный акт сверки по каждому клиенту, так и консолидированный отчет по всем клиентам в рамках одного юридического лица.

ПРИМЕЧАНИЕ Внешний отчет «Акт сверки» находится в файле FileO21 .zip в материалах к данной книге по адресу www.piter.com/download.

Лист загрузки в автомобиль

Перед тем как делать отборку товара на складе, начальник транспортного отдела определяет маршрут каждого водителя-экспедитора. Маршрут определяется в соответствии с теми накладными, которые были подготовлены оператором. Ограничивающими факторами являются:

- вес загрузки автомобиля,
- количество посещаемых автомобилем точек.

После того как маршрут определен, формируется лист загрузки (то есть реестр расходных накладных и список товаров, которые вошли в расходные накладные).

ПРИМЕЧАНИЕ Отчет, выполняющий описанную выше схему, находится в файле FileO22.zip в материалах к данной книге по адресу www.piter.com/download.

Товарный отчет в «Бухгалтерии» по форме «Торговли»

Если вам кажутся тяжеловесными стандартные бухгалтерские отчеты (карточка и анализ счета), то вы можете взять за основу отчет «Ведомость» по остаткам ТМЦ из «Торговли». Отчет имеет режимы показа остатков, режим оборотно-сальдовой ведомости в разрезе товаров и оборотно-сальдовой ведомости в разрезе документов.

ПРИМЕЧАНИЕ Описанный выше отчет находится в файле FileO23.zip в материалах к данной книге по адресу www.piter.com/download.

Ввод на основании с помощью внешнего отчета

Иногда возникает ситуация, когда один документ следует ввести на основании другого, а возможности изменить конфигурацию нет. Например, клиент вместо документа инвентаризации заполнил приходную накладную и спрашивает по телефону, что ему делать. Если бы в конфигурации можно было ввести инвентаризацию на основании приходной накладной, то проблем бы не было.

Можно, конечно, приехать к клиенту и разобраться с проблемой на месте или изменить Конфигуратор дома (в офисе) и послать по почте измененный файл MD, а можно написать внешний отчет, который заменял бы процедуру ввода на основании.

Пример внешнего отчета, который составлялся для одного из клиентов, приведен ниже. Отчет позволяет объединить данные в документе «Приходная накладная» с данными документа «Инвентаризация».

```
Процедура Сформировать()
    Табз = СоздатьОбъект("ТаблицаЗначений");
    Инв = СоздатьОбъект("Документ");
    Инв.НайтиДокумент(Инвентаризация);
    Инв.ВыгрузитьТабличнуюЧасть(Табз);
Приход.ВыбратьСтроки();
    Пока Приход.ПолучитьСтроку() = 1 Цикл
        Табз.НоваяСтрока();
        Табз.Товар = Приход.Товар;
        Табз.ИнвКоличество = Приход.Количество;
    КонецЦикла;
    Табз.Свернуть("Товар","ИнвКоличество,Количество");
    Табз.Сортировать("+Товар");
    Инв.ЗагрузитьТабличнуюЧасть(Табз);
    Инв.Записать ();
КонецПроцедуры
```

«Убивалка 1С»

Этот код позволяет «безболезненно» завершить запущенные приложения «1С». Код будет полезен в случае, когда требуется сделать архив или сделать изменение в структуре данных «1С». Однако если у одного из пользователей есть обрабатывающиеся документы или отчеты, то код работать не будет.

```
Процедура ПросмотрБлокнота() Экспорт
    Если ФС.СуществуетФайл(сокрлп(КаталогИБ()) + "break.fig") = 1
      Тогда
        Если Завершение > 0 Тогда
            Завершение = Завершение - 1;
        Иначе
            Завершение = 3;
       , КонецЕсли;
    '•'';:."  Если Завершение = 0 Тогда
            КомандаСистемы("del" + Симв(34) + сокрлп(КаталогИБ())
                + "break.fig" + Симв(34));
            ЗавершитьРаботуСистемы(0);
            Возврат;
        КонецЕсли;
        Возврат;
    КонецЕсли;
    Если Завершение > 0 'Тогда
        Завершение = Завершение - 1;
        Если Завершение = 0 Тогда
            КомандаСистемы("del" + Симв(34) + сокрлп(КаталогИБ())
              + "break.fig" + Симв(34));
            ЗавершитьРаботуСистемы(0);
            Возврат;
        КонецЕсли;
    КонецЕсли;
КонецПроцедуры
Процедура ПриНачалеРаботыСистемы()
    // Выполняется при старте.
    Если ФС.СуществуетФайл(сокрлп(КаталогИБ()) + "break.fig") = 1
      Тогда
        ЗавершитьРаботуСистемы(0);
        Возврат;
    КонецЕсли;
КонецПроцедуры
ОбработкаОжидания("ПросмотрБлокнота",20);
```

Глава 3
Советы программисту

Пошаговая стратегия работы с клиентом

Вот стандартная схема продаж 1С-услуг:

1. Провести предпроектное обследование.
2. Написать техническое задание на внедрение программы.
3. Внедрить программу.
4. Заключить договор обслуживания.

Каждый из этапов оплачивается отдельно. Согласно этой стратегии заказчик видит результаты работы программиста (и может реально вмешаться в ход внедрения) только на третьем этапе. Вы можете возразить, что клиент сам формулирует и пишет техническое задание. На это я возражу так: техническое задание пишется группой программистов для себя же, а у заказчика просто нет людей, которые бы обладали квалификацией, достаточной для составления технического задания.

Возникает противоречие. С одной стороны, разработчик должен начать строительство программы в соответствии с планом, а с другой стороны, заказчик может повлиять на ситуацию только после*того, как сможет «пощупать» систему своими руками. Выходом из противоречия может стать пошаговая стратегия.

В своей практике я почти не составляю технических заданий, так как их составление — это толчение воды в ступе. Разговор с исполнителем и язык графов позволяют на ходу составлять спецификацию задачи и почти сразу приступать к программированию.

Пошаговая стратегия осуществляется по следующей схеме:

1. Определяется участок предприятия, на котором можно провести внедренческий эксперимент. Это может быть филиал, или удаленный склад, или кондитерский цех, или магазин.
2. На основании полученного опыта программа внедряется на других участках.

Техническое задание при такой стратегии можно составлять устно. Большее значение следует придать поддержанию психологического контакта с заказчиком, а не правильности заключения бумаг. Вместо составления бумаг можно раз в неделю устраивать «разборы полетов» и определять очередной этап работ.

Положительные стороны пошагового подхода:

- По сравнению с традиционным подходом становится быстрее понятно, чего заказчик хочет на самом деле (потому что это не всегда совпадает с тем, что он заявляет как желаемое).
- Инкрементальный подход позволяет на ходу внедрять стандартные схемы работы.
- Инкрементальный подход позволяет избежать проблемы «Даты Ч», к которой программа должна быть внедрена.

Отрицательные стороны пошагового подхода:

- По сравнению с традиционным подходом пошаговый подход имеет больше неопределенности в отношениях программист — заказчик.
- Появляется больший риск наслаивания срочных проблем, которые необходимо решать здесь и сейчас.

Казалось бы, в случае, когда заключается пошаговый договор, программист рискует больше, чем когда работа ведется по традиционной схеме. Однако, начиная сотрудничество, клиент вкладывает ресурсы в покупку новой программы, тратится на обучение, отрабатывает ошибки. В определенный момент времени он становится зависим от команды программистов, так что продолжить сотрудничество для него получается гораздо проще, чем прекратить.

Как описать систему, чтобы ее можно было запрограммировать?

Задача для программиста должна быть сформулирована в терминах программирования. К сожалению, не все понимают этот очевидный принцип. Чтобы задачу можно было запрограммировать, она должна содержать ключевые слова: *константа, справочник, подчиненныйсправочник, документ, периодическаявеличина, начальныйостаток, приход, расход, остаток, регистр, счет*.

Пример переформулирования задачи

Первоначально задача была сформулирована клиентом так: «Требуется учитывать арендные выплаты за пользование холодильниками».

Изменим задачу, чтобы условие включало в себя ключевые слова из лексикона программиста.

Переформулируем задачу «для себя»: «Требуется создать *отчет* "клиент - холодильник - время", который должен строиться на основе *регистра* "Аренда". Регистр должен формироваться *документом* "Акт передачи холодильника в пользование"».

Для проверки правильности понимания задачи перескажем задачу заказчику.

Программист: Я правильно понял Вас, что в «1С» следует сделать *документ,* в который оператор будет заносить данные о том, что холодильник был передан пользователю, и на основании этого документа будет начисляться помесячная аренда?

Клиент: Да, Вы правильно поняли.

Клиент подтвердил правильность рассуждений, значит можнопрограммировать.

Для чего это нужно?

Просите клиента объяснить, для чего ему нужно то, о чем он просит. Обсудите, как он будет использовать плоды вашего труда. Может быть, он неверно представляет себе конечный результат? Попробуйте в уме проиграть различные варианты будущего.

Скажем, клиент просит сделать реестр накладных. Попросите объяснить, для чего он ему нужен. В ходе объяснений может оказаться, что «листочек», который он просит составить, на самом деле является планом поездки экспедитора, и вам нужно организовать работу программы так, чтобы одна накладная не попадала двум разным экспедиторам. Или может оказаться, что заказчик просил сделать не отчет, а документ, по которому бы сверялись фактические отгрузки и начислялась бы заработная плата экспедиторов.

Выяснению «для чего все это нужно» хорошо помогает подбадривающая фраза «это все я хочу, для того чтобы...». Эту фразу вы говорите от имени клиента и просите его продолжить.

«Запрещенные» и «разрешенные» вопросы в метамодели клиента

Под запрещенными и разрешенными вопросами я понимаю не те вопросы, которые человек может или не может задавать, а такие вопросы, которые не ожидаешь или не предполагаешь услышать от собеседника.

Существуют разные «запрещенные» вопросы для разных специалистов в соответствии с спецификой их работы. Например, бухгалтеров интересуют бухгалтерские счета, а работника торгового отдела интересуют неснижаемый остаток, сумма товара на складе, наценка, заработная плата агентов и просроченная задолженность клиентов. Если бухгалтер начинает рассуждать о просроченной задолженности клиента (что явно не входит в специфику его работы), нужно попросить уточнить, что он имеет в виду. И наоборот, вопрос, правильно ли списывается товар со склада, возможен в метамодели бухгалтера, но никак не в метамодели торгового работника.

А что нужно на самом деле?

Что вы ответите на вопрос клиента: «Скажите, уважаемый программист, правильно ли списывается товар со склада?» Я бы ответил так:

— Насколько я понял, Вас интересует, правильно ли списывается товар с 41-го счета? *(Это свое города проверка, понимает ли мой собеседник бухгалтерский язык.)*

— Да нет, меня не интересует бухгалтерская муть... *(Собеседник говорит, что он не бухгалтер.)*

— Хорошо, а зачем тогда Вам нужно знать, правильно ли списывается товар? *(Дело в том, что начальная фраза может звучать только из уст бухгалтера. Не бухгалтер должен сформулировать вопрос иначе.)*

— Я хочу посчитать наценку. *(Ага, понятно: клиент думал об одном, а задал совершенно другой вопрос.)*

— Скажите, а наценка с точностью плюс-минус лапоть устроит, или же следует определять наценку с бухгалтерской точностью? *(Этот вопрос сужает варианты программной реализации.)*

— Меня устроят приблизительные цифры. Я должен знать, что наценка составляет 10 %, а не 5 или 14. *(Последовательными уточняющими вопросами было определено, что клиенту было нужно на самом деле.)*

Регистры или бухгалтерские счета?

Перед проектированием программы следует выяснить, кто главный в информационной системе предприятия. Если главный постановщик задачи — торгаш, то следует программировать «Торговлю». Если львиная часть времени будет уходить на общение с бухгалтером, то следует устанавливать «Бухгалтерию».

Вы также должны иметь в виду, что вы — не единственный в мире 1С-программист, и у вас есть множество конкурентов, которые спят и видят, как подпортить вам жизнь. Оградить заказчика от встреч с конкурентами невозможно. Представьте, что один из конкурентов придет и покажет заказчику-торгашу, какие в «Торговле» бывают графики, в то время как вы предоставляете ему скучные таблицы цифр. Вполне возможно, что начальник «купится» на картинки и решит, что вы — несолидный партнер.

Возникает непреодолимый культурный барьер понимания бухгалтерии торгашом и торговли бухгалтером. Бухгалтер рано или поздно захочет увидеть оборотно-сальдовую ведомость по какому-либо счету. Убедить бухгалтера в том, что торговля — это «круто», просто невозможно. Так же как невозможно убедить его в том, что переброска документов из «Торговли» в «Бухгалтерию» — это то, о чем он мечтал всю жизнь.

В вопросе, стоит ли внедрять «Торговлю» или «Бухгалтерию», следует определить уровень бардака на предприятии. Попробуйте выяснить, насколько жесткая война идет между торговым отделом и бухгалтерией. Если война есть, то лучше внедрять обе конфигурации. В этом случае вы избежите ситуации, когда у главного бухгалтера случится инфаркт из-за того, что данные в «закрытом месяце» внезапно изменились, так как работник торгового отдела откатил дату редактирования и убрал налог с продаж в одном из расходных документов.

Если же предприятие выросло из коротких «информационных штанишек», если его руководство понимает роль дисциплины в информационных системах и если изменения задним числом для такого предприятия — нонсенс, то я бы дал совет программировать информационную систему в «Бухгалтерии». Она гибче и пластичнее, чем «Торговля», к тому же все управленческие задачи, стоящие перед «Торговлей», можно запрограммировать в «Бухгалтерии».

Выясните отличия схемы, действующей на предприятии, от стандартной

Бывает так, что бухгалтеры не знают, как они работают. Работники сменяют друг друга, и некогда данные им инструкции теряют первоначальный смысл. Часть инструкций забывается, а работа ведется как получится. Часть бухгалтеров может объяснить, что они делают и что им нужно делать, но не могут объяснить, как их работа соотносится с работой других.

Приведу в качестве примера диалог программиста и бухгалтера.

Программист: Объясните, как у вас считается себестоимость.

Бухгалтер: Не знаю. Себестоимость считает плановый отдел.

Программист: Хорошо, а не могли бы Вы объяснить, какими этапами из приведенной схемы Вы занимаетесь:

62<-43<-20<г-10<-60.

Бухгалтер: У нас не такая схема. У нас вот такая схема:

20 <— 10 (производство) <— 16 <— 10 (сырье) <— 60.

Программист: Хорошо, а что идет после счета 20?

Бухгалтер: Не знаю.

Программист: А зачем вам нужен счет 16?

Бухгалтер: Дело в том, что на складе сырья продукция хранится в виде материала, а на складе готовой продукции она хранится в литрах.

Программист: А почему вы не делаете проводки 43 <— 20 <— 10?

Бухгалтер: Я не могу делать такой проводки, потому что вся продукция у меня учитывается в литрах (!).

Вот такой получился разговор. Вы что-нибудь поняли? Я — нет.

Если вы будете слушать такого бухгалтера, то вы никогда не придете к правильному решению. Стандартная схема превращения материала в готовую продукцию говорит о том, что следует делать следующие проводки: 43 <— 20 <— 10. Эту схему и следует внедрять, если никакой другой схемы нет, даже в том случае, когда клиент не совсем представляет, как все будет выглядеть.

Предлагайте свои модели поведения. Лучше пусть будет ваша модель поведения, чем никакой. Если тот, кто может повлиять на ситуацию, ничего не может привнести полезного в вашу модель, то его следует убедить в том, что ваша модель — лучшая.

У принципа «делать, как знаю» есть и оборотная сторона. Попытка уложить факты в прокрустово ложе своих представлений может закончиться печально для проекта, поскольку вы не можете учесть все существующие факторы. Поэтому следует проверять и совершенствовать свои модели. Лучший способ совершенствовать модели — общаться с профессионалами.

Советы программисту

- Программируйте только у клиента, не берите работу домой или в офис.
- Делайте работу на одном дыхании в состоянии высшего подъема. Любая задача должна быть выполнена в этот же день, иначе она не будет выполнена никогда.
- Печатайте быстрее. Изучите навык слепой печати. Научившись быстро печатать, человек не отвлекается на клавиатуру и освобождает голову для обдумывания мыслей.
- Сводите задачи к простым. Запутанные схемы приводите к описанным заранее моделям. Предлагайте типовую схему решения проблемы. Выявляйте расхождения между типовой схемой и реальной ситуацией у клиента.
- Выявите заблуждения клиента.
- Делайте только то, что необходимо, а не то, что озвучил заказчик.
- Пользуйтесь административной властью начальников клиентов.

- Если можно не программировать — не программируйте. Старайтесь свести свое вмешательство в программу к минимуму. Если отчет можно сделать внешним, то его не стоит записывать в структуру программы.
- Используйте графы при описании задачи.
- Делайте печатные формы и формы документов и отчетов красивыми.
- Делайте результаты своей работы передаваемыми другому специалисту.
- Не посвящайте одному заказчику более четырех часов в день, иначе он начнет считать, что вы постоянный работник.

Что делать с тем, что уже сделано до вас?

Нередки случаи, когда у заказчика уже установлена одна из версий программы, которую настраивали другие программисты. Что делать? Предлагать ли клиенту свою программу или взять за основу то, что было сделано другими?

Ломать — не строить, однако в пылу разрушения не забудьте, что, во-первых, кое-что вы просто обязаны сохранить (как минимум — остатки товаров, задолженности клиентов, информацию справочников). Подумайте о том, что ваши предшественники уже решили многие проблемы и обошли многие подводные камни, с которыми вам еще только предстоит столкнуться. К старой программе пользователи привыкли, поэтому вы можете встретить сопротивление, если будете что-то существенно изменять. Другой причиной не ломать сделанное может быть то, что сопровождение программы — операция менее трудоемкая, чем ее написание «с нуля». Кроме того, поддержкой может заниматься и менее опытный специалист из вашей команды, а перестройка по плечу только профессионалу высокого уровня. Наконец, вы рискуете сделать новую систему еще хуже, чем предыдущая.

Существует еще одна причина не делать крупных реорганизаций. Реорганизации — это отвлечение серьезных сил от других проектов. Если вы ведете одновременно несколько предприятий, то вам придется работать в режиме пиковых нагрузок. Клиент, у которого идет внедрение новой программы, захочет лицезреть вас каждый день. Стоит нескольким клиентам пожелать этого одновременно — и ваша клиентская сеть затрещит по швам.

И наконец, еще один довод в пользу того, чтобы сохранить созданное другими. Заказчик, как правило, понимает термин «обслуживание» в смысле «чтобы все работало», а не в смысле «сломать то, что уже есть, и сделать по-другому, чтобы было за что выставить солидный счет», как понимает его программист.

Возможно, вы возразите, что предыдущий программист был совсем дурак и от него было больше вреда, чем пользы. Откровенно говоря, я таких случаев не встречал. У другого всегда есть чему поучиться и есть что позаимствовать.

Глава 4
Язык программирования «1С»

В этой главе приводится краткий обзор языка программирования «1С». Его полное описание можно найти, например, в книге «1С:Предприятие, версия 7.7. Описание встроенного языка», которая содержит 900 страниц и продается вместе с продуктами фирмы «1С». Дополнительную информацию вы можете найти в Интернете.

Агрегатные типы данных

В системе «1С» поддерживаются базовые и агрегатные типы данных. К базовым типам относятся числа, строки и даты. Агрегатные типы данных — это специализированные типы, предназначенные для работы с объектами «1 С.Предприятия».

- Константа — средство работы с постоянными (или условно постоянными) значениями. В константах хранится информация, которая не изменяется или изменяется достаточно редко, например название организации или почтовый адрес. Перечень констант, доступный в конкретной конфигурации, их названия и типы определяются в Конфигураторе.
- Справочник — средство для ведения списков однородных элементов данных. Помимо наименования элементов, данных списки могут содержать дополнительную информацию. Физическим аналогом справочника является картотека. Каждая карточка — это элемент справочника, а сведения, заносимые в карточку, являются реквизитами справочника. Перечень справочников, доступных в конкретной конфигурации, их названия и реквизиты определяются в Конфигураторе.
- Перечисление — средство работы с элементами данных, список возможных значений которых жестко задан (например, для перечисления Булево можно задать возможные значения: Да, Нет). В отличие от справочников, списки значений в перечислении задаются в процессе их создания в Конфигураторе и при выполнении задачи не могут быть изменены. Состав перечислений, доступных в конкретной конфигурации, их названия и допустимые значения определяются в Конфигураторе.
- Документ — средство для ввода первичной информации о совершаемых хозяйственных операциях. Перечень документов, доступных в конкретной

конфигурации, их названия, реквизиты и другие свойства определяются в Конфигураторе.

- Запрос — средство для выполнения обращения к документам, регистрам, справочникам и журналам расчетов с целью получения сводной информации при формировании выходных отчетов. В программных модулях допускается создавать произвольное число объектов типа Запрос при помощи вызова системной функции СоздатьОбъект.
- Текст — средство работы с текстовыми документами. В программных модулях допускается создание произвольного числа объектов типа Текст при помощи вызова системной функции СоздатьОбъект.
- Таблица — средство работы с таблицами (отчетами). В программных модулях допускается создание произвольного числа объектов типа Таблица при помощи вызова системной функции СоздатьОбъект.
- СписокЗначений — средство для создания списка значений каких-либо данных, позволяющего в дальнейшем сортировать и выбирать нужные значения из списка. При добавлении в диалоговых формах полей типа Список или Поле со списком система автоматически создает объекты СписокЗначений, доступ к которым в языке возможен по идентификатору поля. В программных модулях допускается создание произвольного числа объектов типа СписокЗначений при помощи вызова системной функции СоздатьОбъект.
- ТаблицаЗначений — средство для создания списка значений каких-либо данных, позволяющего в дальнейшем сортировать и выбирать нужные значения из списка. При добавлении в диалоговых формах полей типа Список или Поле со списком система автоматически создает объекты СписокЗначений, доступ к которым в языке возможен по идентификатору поля. В программных модулях допускается создание произвольного числа объектов типа СписокЗначений при помощи вызова системной функции СоздатьОбъект.
- Картинка — средство для работы с графическими файлами. При добавлении в диалоговых формах и в таблицах полей типа Картинка система автоматически создает объекты Картинка, доступ к которым в языке возможен по идентификатору поля. В программных модулях допускается создание произвольного числа объектов типа Картинка при помощи вызова системной функции СоздатьОбъект.
- Периодический — средство для работы с периодическими реквизитами справочников и периодическими константами. В программных модулях допускается создание произвольного числа объектов типа Периодический при помощи вызова системной функции СоздатьОбъект.
- ФС — средство для работы с дисковыми файлами непосредственно из встроенного языка системы «1С:Предприятие». В программных модулях допускается создание произвольного числа объектов типа ФС при помощи вызова системной функции СоздатьОбъект. Кроме того, в глобальном контексте по умолчанию существует один уже созданный объект этого типа с именем ФС (имя объекта совпадает с названием агрегатного типа данных).

- XBase — средство для работы с файлами баз данных DBF-формата непосредственно из встроенного языка системы «1С:Предприятие». В программных модулях допускается создание произвольного числа объектов типа XBase при помощи вызова системной функции СоздатьОбъект.

Следующие типы данных доступны только при наличии компоненты «Бухгалтерия».

- ПланСчетов (служебный тип данных) предназначен для идентификации Плана счетов, созданного в метаданных. Тип значения ПланСчетов не поддерживает никаких данных в информационной базе, а список возможных значений этого типа данных определен планами счетов, созданными в конфигурации. Для получения значения такого типа данных используется глобальный атрибут ПланыСчетов, который имеет, в свою очередь, набор атрибутов типа ПланСчетов, соответствующих имеющимся в конфигурации планам счетов.
- ВидСубконто (служебный тип данных) предназначен для идентификации вида субконто, созданного в метаданных. Список возможных значений этого типа данных определен видами субконто, созданными в конфигурации. Для получения значения такого типа используется глобальный атрибут ВидыСубконто, который имеет, в свою очередь, набор атрибутов типа ВидСубконто, соответствующих имеющимся видам субконто. Кроме того, глобальный атрибут ВидыСубконто имеет методы для обхода всех существующих видов субконто.
- Операция — средство для работы из встроенного языка с данными бухгалтерских операций и проводок, формируемых документом. Так как проводки в системе «1С:Предприятие» принадлежат операциям, то управление и операциями, и проводками выполняется объектом Операция.
- БухгалтерскиеИтоги — средство для организации доступа к бухгалтерским итогам в различных разрезах, за различные периоды и с разной степенью детализации. При наличии в системе «1С:Предприятие» компоненты «Бухгалтерия» система автоматически реализует специальный механизм работы с бухгалтерскими итогами. Данный механизм обеспечивает хранение, динамический пересчет бухгалтерских итогов и их извлечение средствами встроенного языка. Система хранения бухгалтерских итогов поддерживается системой «1С:Предприятие» автоматически на основе существующих планов счетов. При редактировании планов счетов — в Конфигураторе или при работе с системой «1С:Предприятие» — для счета может быть установлен ряд свойств, влияющих на организацию хранения бухгалтерских итогов: это признаки ведения валютного и количественного учета, а также включение аналитического учета по субконто. Изменение бухгалтерских итогов может производиться только проводками бухгалтерских операций.

Следующий тип данных доступен только при наличии компоненты «Оперативный учет».

- Регистры — средство накопления сводной информации. Регистры состоят из измерений, ресурсов и реквизитов. Регистры позволяют определять начальный, конечный остаток и обороты ресурсов, а также производить фильтрацию данных по реквизитам. Регистры могут изменяться только из документов. Регистр отчасти является аналогом бухгалтерских итогов.

Каждый агрегатный тип данных, как правило, имеет набор атрибутов и методов. Атрибуты по свойствам напоминают переменные, то есть им можно присваивать значения и считывать их. Методы — это те действия, которые может выполнять агрегатный тип данных. Методы могут иметь или не иметь возвращаемое значение. Если метод имеет возвращаемое значение, то он может размещаться в правой части оператора присваивания в выражениях и в описании фактических параметров других вызываемых методов, процедур или функций.

Типичная последовательность работы с объектом агрегатного типа выглядит следующим образом.

1. С помощью функции СоздатьОбъект создается объект агрегатного-типа и какой-либо переменной присваивается ссылка на него.
2. Объект позиционируется на нужном элементе данных.
3. Манипуляции с объектом агрегатного типа производятся через вызовы методов и обращения к его атрибутам.

В том случае, если объект агрегатного типа больше не нужен, он может быть отсоединен от переменной посредством присваивания ей какого-либо значения базового типа (например, числа 0).

Пример использования функции СоздатьОбъект:

```
// Создаем объект.
Тов = СоздатьОбъект("Справочник.Номенклатура");
// Производим манипуляции с объектом.
Тов.ВыбратьЭлементы();
Пока Тов.ПолучитьЭлемент() = 1 Цикл
    Сообщить("Наименование товара " + Тов.Наименование);
КонецЦикла;
```

Отсоединение объекта выполнять необязательно.

Справочники

Справочник — это агрегатный тип данных, предназначенный 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.

Ниже приведены основные методы работы с элементами справочников.

Метод НайтиЭлемент

Метод НайтиЭлемент производит поиск элемента справочника. Рассмотрим пример записи цены в карточку товара:

```
Тов = СоздатьОбъект("Справочник.Номенклатура");
Тов.НайтиЭлемент(Запрос.Тов);
Тов.Цена = 0;
Тов.Записать();
```

Казалось бы, получилось «масло масляное»: стоит ли находить элемент, если он и так найден? Однако операция Записать применима только к тем объектам, которые созданы командой СоздатьОбъект. То есть следующий код будет некорректным:

```
Запрос.Тов.Цена = 0/
.Запрос.Тов.Записать();
```

Поиск элемента справочника

Для поиска элемента справочника используются следующие методы:

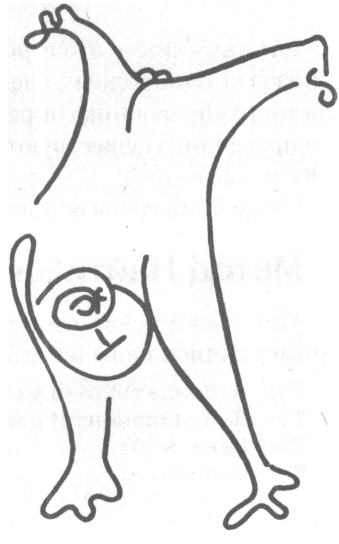
- НайтиЭлемент *(<Элемент>)*,
- НайтиПоКоду *(<Код>, <ФлагПоиска>)*,
- НайтиПоНаименованию *(<Наименование>, <Режим>, <ФлагПоиска>)*,
- НайтиПоРеквизиту *(<Реквизит>, <Значение>)*.

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

Пример поиска элемента справочника:

```
Функция ПоискЭлемента(Справ,Наим)
Спр = СоздатьОбъект("Справочник");
Спр.Вид(Справ);
Если Спр.НайтиПоНаименованию(Наим,0) = 0 Тогда
    Спр.Новый();
    Спр.Наименование = Наим;
    Спр.Записать();
КонецЕсли;
Возврат Спр.ТекущийЭлемент();
КонецФункции
```

Выборка элементов справочника



Для обработки элементов справочника можно использовать следующую последовательность команд:

```
Тов =
СоздатьОбъект("Справочник.Товары");
Тов.ВыбратьЭлементы();
Пока Тов.ПолучитьЭлемент() = 1 Цикл
    Если Тов.ЭтоГруппа() = 1 Тогда
        // Если это группа товаров,
   // то пропускаем.
        Продолжить;
    КонецЕсли;
    Сообщить("Товар " + Тов.Код + " " +
        Тов.Наименование);
КонецЦикла;
```

Другие полезные команды, которые связаны с выборкой элементов справочника:

- Спр .ИспользоватьВладельца *(<Владелец>),*
- Спр.ИспользоватьРодителя *(<Группа>).*

Работа с группами элементов

Справочник может иметь иерархическую структуру. Число уровней иерархии определяется в поле Кол-во уровней в окне редактирования. Наиболее важные функции:

- ЭтоГруппа () — возвращает 1, если текущий элемент справочника — группа;
- ПринадлежитГруппе *(<Группа>)* — возвращает 1, если текущий элемент принадлежит указанной группе;
- Уровень () — возвращает номер уровня текущего элемента.

Добавление нового элемента и группы в справочник

Для добавления нового элемента в справочник используется метод Новый:

```
Спр = СоздатьОбъект("Справочник.Номенклатура");
Спр.Новый();
Спр.Наименование = "Новый элемент";
Спр.Записать();
```

Для добавления новой группы в справочник используется метод НоваяГруппа:

```
Спр.НоваяГруппа();
Спр.Наименование = "Новая группа элементов";
Спр.Записать();
```

Работа с подчиненными справочниками

Подчиненный справочник позволяет организовать отношение «один ко многим». Каждому элементу справочника-владельца будет соответствовать свой набор элементов подчиненного справочника. Например, один элемент справочника Товар может иметь несколько связей с элементами справочника Цены.

Если один справочник подчинен другому, то это следует указать в поле подчинен окна редактирования.

Для обработки подчиненного справочника можно использовать следующий код:

```
Тов = СоздатьОбъект("Справочник.Товары");
Цен = СоздатьОбъект("Справочник.Цены");
Тов.ВыбратьЭлементы();
Пока Тов.ПолучитьЭлемент() = 1 Цикл
    Сообщить("Товар " + Тов.Код + " " + Тов.Наименование) ;
    Цен.ИспользоватьВладельца(Тов.ТекущийЭлемент());
    Цен.ИспользоватьДату(ТекущаяДата());
    Цен.ВыбратьЭлементы();
    Пока Цен.ПолучитьЭлемент() = 1 Цикл
        Сообщить("Цена товара " + Цен.Цена + " " + Цен.ВидЦены);
```

```
    КонецЦикла;
КонецЦикла;
```

Программа показывает список цен (подчиненный справочник Цены) для каждого товара справочника Товары.

Документы

Документы в системе «1С:Предприятие» используются для ввода, просмотра и корректировки информации о совершаемых хозяйственных операциях. У любого документа есть три обязательных реквизита: ДатаДок, ВремяДок, НомерДок. Дата и время — наиболее важные характеристики документов, так как позволяют устанавливать строгую временную последовательность совершения операций. Форма документа редактируется при нажатии кнопки Форма.

Реквизиты шапки документа задаются в списке Реквизиты шапки. Реквизиты табличной части задаются в списке Реквизиты табличной части.

Проведение документа

Для того чтобы документ формировал проводки, необходимо создать хотя бы один план счетов и установить флажки Разрешить проведение документа и Бухгалтерский учет (Конфигуратор • Конфигурация • Открыть • Конфигурацию • Документ • <Конкретный документ> • Редактировать). Если документ будет формировать движение регистров, следует установить флажок Оперативный учет.

Пример формирования проводок документом:

```
Операция.НоваяПроводка();
Операция.СодержаниеПроводки = "Поступили товары";
Операция.НомерЖурнала = "ТВ";
Операция.Дебет.Счет = СчетПоКоду("41.1");
Операция.Дебет.Номенклатура = Товар;
Операция.Кредит.Счет = СчетПоКоду("60.1") ;
Операция.Кредит.Контрагенты = Контрагент;
Операция.Кредит.Договоры = Договор;
Операция.Количество = Количество;
Операция.Сумма = УчетнаяСтоимость;
Операция.Записать();
```

Пример формирования движений регистров при проведении документа:

```
Процедура Кеги(Док) Экспорт
Док.ВыбратьСтроки();
ДокВид = Док.Вид();
Пока Док.ПолучитьСтроку() = 1 Цикл
     Рег = Док.Регистр.Кеги;
     Если (Док.Товар.ВидТовара о Перечисление.ВидыТоваров.Тара)
       Тогда
          // Если это не тара, то продолжить.
          Продолжить;
     КонецЕсли;
     Рег.Клиент = Док.Клиент;
     Рег.Товар = Док.Товар;
     Рег.Количество = Док.Количество;
```

```
    Если ДокВид = "РасходнаяНакладная" Тогда
        Per.ДвижениеПриходВыполнить    ();
    ИначеЕсли ДокВид = "ПриходнаяНакладная" Тогда
        Per.ДвижениеРасходВыполнить();
    КонецЕсли;
КонецЦикла;
КонецПроцедуры
```

Выборка документов

Для выборки документов можно использовать следующую последовательность команд:

```
Процедура Сформировать()
Док = СоздатьОбъект("Документ");
Док.ВыбратьДокументы();
Пока Док.ПолучитьДокумент() = 1 Цикл
    Сообщить("Документ - " + Док.Вид() + " " + Док.ДатаДок + " "
      + Док.НомерДок);
    Док.ВыбратьСтроки();
    Пока Док.ПолучитьСтроку() = 1 Цикл
        Сообщить("Товар - " + Док.Товар);
    КонецЦикла;
КонецЦикла;
КонецПроцедуры
```

Создание нового документа

Для ввода документа используется метод Новый. Для ввода новой строки документа используется метод НоваяСтрока. Для записи документа используется метод Записать. Для проведения документа используется метод Провести.

Пример создания нового документа:

```
Процедура Сформировать()
Док * СоздатьОбъект("Документ.РасходнаяНакладная");
Док.Новый();
Док.ДатаДок = ТекущаяДата();
Табз.ВыбратьСтроки();
Пока Табз.ПолучитьСтроку() = 1 Цикл
    Док.НоваяСтрока();
    Док.Товар = Табз.Товар;
    Док.Количество = Табз.Количество;
    Док.Цена = Док.Товар.Цена1;
КонецЦикла;
Док.Записать();
Док.Провести();
КонецПроцедуры
```

Использование печатных форм (объект Таблица)

Для создания печатных форм и для ввода табличных данных используется объект Таблица. Таблицы могут располагаться в разделе Общие таблицы и в фор-

мах визуальных объектов системы. Форма может иметь одну таблицу для ввода и много таблиц для вывода.

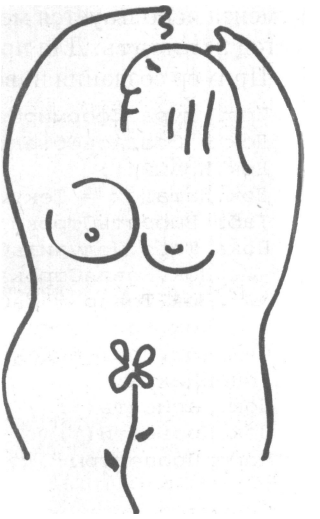
Таблица является шаблоном для вывода данных. В свойствах каждой ячейки можно задать ее тип: Текст, Выражение, Шаблон и Фиксированный шаблон. Текст печатается так, как он задан в Конфигураторе. Выражение вычисляется, и в ячейке записывается результат. Шаблон — это текст, в котором может присутствовать выражение, которое задается в квадратных скобках, например "Сумма равна [ПечСумма] ".

Пример использования объекта Таблица:

```
Таб = СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("Накладная");
Таб.ВывестиСекцию("Шапка");
Док.ВыбратьСтроки();
Пока Док.ПолучитьСтроку() = 1 Цикл
    Таб.ВывестиСекцию("Строка|Тело");
    Таб.ПрисоединитьСекцию("Строка|НП");
КонецЦикла;
Таб.ВывестиСекцию("Подвал");
Таб.Показать();
```

В Конфигураторе предусмотрен легкий для понимания конструктор печатных форм: Конфигуратор • Конструкторы • Макет отчета. Советую начать конструирование печатных форм с него.

Объект СписокЗначений



Объект СписокЗначений применяется для создания динамических списков (не сохраняемых в БД), которые могут отображаться в диалоговых формах (элементы Список и Поле со списком). Основные возможности использования списка значений:

- выбирать значение из заранее составленного списка;
- добавлять новые элементы в список значений;
- сортировать список;
- удалять элементы списка.

Элемент списка содержит три поля: Значение, Представление и Пометка. Значения могут быть данными любого типа, представление же всегда имеет тип Строка. В форме диалога отображается представление (если оно задано). Пометка означает, помечено данное значение или нет.

Основные функции работы со списком следующие:

- ДобавитьЗначение *(<Значение>,<Строка>)*,
- ПолучитьЗначение *(<Позиция>,<Переменная>)*,
- Получить *(<Строка>)*,

- УдалитьЗначение *(<Позиция>, <Количество>),*
- УдалитьВсе (),
- Сортировать *{<Направление>)*,
- Принадлежит *(<Значение>),*
- ИзСтрокиСРазделителями *(<Строка>),*
- ВСтрокуСРазделителями (),
- РазмерСписка (),
- НайтиЗначение *(<Значение>).*

Пример использования списка значений:

```
// Процедура используется в обработке подбора таблицы значений.
Процедура ОбработкаПодбора(ЗначПод)
Если (ЗначПод.Вид() = "Номенклатура") Тогда
    Если ЗначПод.ЭтоГруппа() = 0 Тогда
        ВыбТовары.ДобавитьЗначение(ЗначПод);
        ВыбТовары.ТекущаяСтрока(ВыбТовары.РазмерСписка());
        Форма.УдалитьСтрокуТовара.Доступность(1);
    КонецЕсли;
КонецЕсли;
КонецПроцедуры
```

ВыбТовары — это объект типа Список, который находится на форме диалогового окна отчета.

Чтобы определить, входит ли выбранный элемент в список значений, можно использовать следующий код (ВыбТовары — список значений):

```
Функция Товар(Тов)
Если ВыбТовары.РазмерСписка() = 0 Тогда
    Возврат 1;
КонецЕсли;
Если ВыбТовары.НайтиЗначение(Тов) <> 0 Тогда
    Возврат 1;
КонецЕсли;
Возврат 0;
КонецФункции
```

Объект ТаблицаЗначений

Объект ТаблицаЗначений применяется для создания динамических массивов (не сохраняемых в БД), которые могут отображаться в диалоговых формах (элемент Таблица значений). Таблицазначений создается с помощью функции СоздатьОбъект ("ТаблицаЗначений[11]"), либо визуально при добавлении на форму элементов Таблица значений.

Основные функции работы с таблицей значений:

- НоваяКолонка*(<Идентификатор>,<Тип>,<Длина>, <Точность>, <Заголовок>,<Ширина>,<Формат>,<Положение>*);
- НоваяСтрока ();
- УдалитьСтроку *{<НомерСтроки>);*

- УдалитьСтроки ();
- ПолучитьСтрокуПоНомеру *(<НомерСтроки>);*
- ВыбратьСтроки () — открыть выборку строк из таблицы значений;
- ПолучитьСтроку () — получить следующую строку из выборки;
- Сортировать *(<Колонки>)* — в качестве параметра используется строка, содержащая список идентификаторов или номеров колонок, разделенных запятой, по которым выполняется сортировка строк таблицы значений;
- Свернуть *(<ГрупКолонки>, <СумКолонки>)* — первый параметр (строка *ГрупКолонки)* содержит список идентификаторов или номеров колонок, разделенных запятой, по которым выполняется группировка строк таблицы значений (то есть если в группируемых колонках имеется несколько строк с одинаковыми значениями, то в результате свертки останется одна такая строка); второй параметр (строка *СумКолонки)* содержит список идентификаторов или номеров колонок, разделенных запятой, в которых выполняется суммирование значений строк таблицы значений (то есть если группируется несколько строк, то значения суммируемых колонок будут складываться).

Пример инициализации таблицы значений:

```
Табз=СоздатьОбъект("ТаблицаЗначений");
// Полное определение реквизитов колонки
Табз.НоваяКолонка("Товар", "Строка", 20,, "Наименование товара",
  20);
// Можно делать сокращенное определение реквизитов колонки.
Табз.НоваяКолонка("Цена");
Табз.НоваяКолонка("Количество");
Док.ВыбратьСтроки();
Пока Док.ПолучитьСтроку() = 1 Цикл
    // Ввод новой строки таблицы значений
    Табз.НоваяСтрока();
    Табз.Товар = Док.Товар;
    Табз.Цена = Док.Цена;
    Табз.Количество = Док.Количество;
КонецЦикла;
```

Пример показывает, как можно создавать колонки и строки таблицы значений.

Выгрузка запроса в таблицу значений

В приведенном ниже примере результаты запроса записываются в таблицу значений:

```
Перем Запрос, ТекстЗапроса, Таб;
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса = "//{{ЗАПРОС(Сформировать)
 IПериод с ВыбНачПериода по ВыбКонПериода;
 Iдок = Документ.рн.ТекущийДокумент, Документ.пн.ТекущийДокумент;
 ITов = Документ.рн.Товар, Документ.пн.Товар;
 IКол = Документ.рн.Количество, Документ.пн.Количество;
```

```
|Акц = Документ.рн.Акциз, Документ.пн.Акциз;
|ВидОп = Документ.рн.ВидОперации, Документ.пн.ВидОперации;
|Функция КолСумма = Сумма(Кол);
|Функция АкцСумма = Сумма(Акц);
|Группировка док;
|Группировка Тов без групп;
|"//}}ЗАПРОС

Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
    Возврат;
КонецЕсли;
// Выгрузка результатов запроса в таблицу значений
Табз = СоздатьОбъект("ТаблицаЗначений");
Запрос.выгрузить(Табз);
Табз.НоваяКолонка("Акциз");
Табз.ВыбратьСтроки();
Пока Табз.ПолучитьСтроку() = 1 Цикл
    Табз.Акциз = "" + Табз.Тов.ГруппаАкциза;
    Сообщить("Товар " + Табз.Тов);
КонецЦикла;
```

После того как результаты запроса выгружены в таблицу значений, программист может добавлять в таблицу значений новые колонки, сортировать строки таблицы и сворачивать данные. Использование таблицы значений может значительно упростить программу.

Выгрузка итогов по регистру в таблицу значений

Средства языка «1С» позволяют выгружать в таблицу значений итоги регистра, например, так:

```
Per = СоздатьОбъект("Регистр.Товары");      |
ТабЗнач = СоздатьОбъект("ТаблицаЗначений");

// Выгрузка итогов в таблицу значений
Per.ВыгрузитьИтоги(ТабЗнач);
ТабЗнач.ВыбратьСтроки();
Пока ТабЗнач.ПолучитьСтроку() = 1 Цикл
    Сообщить("Товар " + ТабЗнач.Товар +
        " на складе " + ТабЗнач.Товар +
        " кол-во: " + ТабЗнач.Количество);
КонецЦикла;
```

Сортировка табличной части документа при помощи таблицы значений

Ниже приведен пример сортировки табличной части документа при помощи таблицы значений. Для сортировки необходимо выгрузить данные табличной части документа в таблицу значений, добавить в таблицу значений новую колонку, по которой будет происходить сортировка, заполнить колонку значениями, по которым будет происходить сортировка, отсортировать таблицу и загрузить ее в документ.

```
Процедура Сортировка()
Табз = СоздатьОбъект("ТаблицаЗначений");
ВыгрузитьТабличнуюЧасть(Табз);
Табз.НоваяКолонка("Родитель");
Табз.ВыбратьСтроки();
Пока Табз.ПолучитьСтроку() = 1 Цикл
    // Предполагается, что в табличной части документа
    // существует реквизит "Товар".
    Табз.Родитель = "" + Табз.Товар.Родитель;
КонецЦикла;
Табз.Сортировать("+Родитель");
ЗагрузитьТабличнуюЧасть(Табз);
КонецПроцедуры
```

Вы можете использовать приведенный код в качестве шаблона манипуляции данными табличной части документа, например, когда требуется печатать строки в документе в определенном порядке или сворачивать строки с одинаковыми товарами.

Как получать управление других форм?

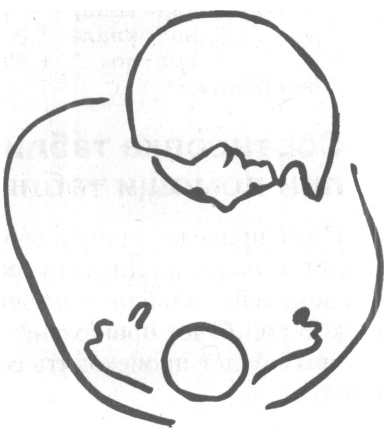
В первый раз вы столкнетесь с понятием «контекст», когда начнете анализировать модуль проведения документов. Возможно, оно будет употреблено в такой конструкции:

```
ПроцедураДвойныеСтроки(Контекст) ;
```

В процедуру, обрабатывающую двойные строки, передается управление документом, для того чтобы процедура, находящаяся вне модуля, могла проводить, записывать, удалять строки или менять реквизиты.

Каждый объектно-ориентированный язык позволяет обращаться к одному объекту из другого. Не является исключением и «1С», где обращение происходит через ссылку на объект, который называется Контекст.

Как открыть форму и оставить ниточки управления у себя?



В справке по 1С-методам находится раздел, посвященный методу ОткрытьФорму. Первый параметр команды используется для заголовка открытой формы, а второй — для передачи параметров:

```
конт = Форма.Параметр;
```

В переменной конт будет переданный параметр. Если конт — это контекст, тогда его можно использовать для доступа к реквизитам документа, методам справочника, объектам на форме и т. д.:

```
Процедура Сформировать()
  конт = "";
ОткрытьФорму("Отчет.ОбработкаСклада",
```

```
  конт);
Если типЗначения(к) = 100 Тогда // Проверяем, открыта ли форма.
    конт.Склад = Константа.ОсновнойСклад;
    конт.Фирма = Фирма;
    конт.Дата_ = ДатаДок;
    конт.форма.Обновить(1);
КонецЕсли;
КонецПроцедуры
```

Процедура устанавливает реквизиты в открываемой форме.

Как открыть форму и передать ей бразды правления?

Для того чтобы передать управление другой форме (открывающая форма руководится открываемой формой), можно использовать следующую конструкцию:

```
// Открывающая Форма
Процедура Сформировать()
ОткрытьФорму("Обработка.ОбработкаСклада", Контекст);
КонецПроцедуры
// Открываемая форма
Процедура ПриОткрытии()
Конт = Форма.Параметр;
Если типЗначения(конт) = 100 Тогда
    pi = Конт.зн1/
    р2 = Конт.зн2;
    р3 = Конт.зн3;
    контА.Форма.Закрыть(0);
КонецЕсли;
КонецПроцедуры
```

С помощью описанного выше механизма в «1С» организованы процедуры Подбор и ВводНаОсновании.

Как передать данные из одной формы в другую?

Допустим, требуется написать обработку, в которую передавались бы данные, но при этом не было бы привязки ни к существующим документам, ни к определяемым в глобальном модуле переменным.

В открывающей форме для этого следует написать следующую процедуру:

```
Процедура Сформировать()
ПередаваемаяПеременная = "Привет";
ОткрытьФорму("Отчет.ПринимающийОтчет", ПередаваемаяПеременная);
КонецПроцедуры
```

В принимающей форме необходимо использовать процедуру ПриОткрытии:

```
Процедура ПриОткрытии()
Если ПустоеЗначение(форма.Параметр) = 1 Тогда
    Возврат; // Ничего не передали
иначе
    к = форма.Параметр;
    сообщить();
КонецЕсли;
КонецПроцедуры
```

Если из одной формы в другую требуется передать набор значений, то можно использовать список значений:

```
// Открывающая форма
Процедура Сформировать()
список = СоздатьОбъект("СписокЗначений");
список.ДобавитьЗначение(зн1);
список.ДобавитьЗначение(зн2);
список.ДобавитьЗначение(зн3);

список.ДобавитьЗначение(ЗНN);
ОткрытьФорму("Отчет.ОченьНужный",  список);
КонецПроцедуры
// Открываемая форма
Процедура ПриОткрытии()
список = форма.Параметр;
Если ПустоеЗначение(список) = 1 Тогда
    Возврат;
конецЕсли;
Если ТипЗначенияСтр(список) = "СписокЗначений" Тогда
    п1 = список.ПолучитьЗначение(1);
    п2 = список.ПолучитьЗначение(2);
    п3 = список.ПолучитьЗначение(3);

    nN = список.ПолучитьЗначение(N);
иначе

КонецЕсли;
КонецПроцедуры
```

В качестве параметров можно передавать имя вызывающей формы, флаги, имена процедур, которые необходимо запускать в принимающей форме.

Применение запросов

ПРИМЕЧАНИЕ Не следует путать запрос в программировании с бухгалтерским запросом.

Запрос создается функцией СоздатьОбъект ("Запрос"). Для выполнения запроса используется метод Выполнить *(<ТекстЗапроса>)*, который возвращает 1, если запрос выполнился. *<ТекстЗапроса>* — это строковое выражение на языке запросов.

Пример использования запроса, который анализирует расходные накладные и показывает количество отгрузок со складов для каждого товара:

```
НашЗапрос = СоздатьОбъект("Запрос");
ТекстЗапроса = "//{{ЗАПРОС(Сформировать)
|СКЛАД = Документ.РасходнаяНакладная.Склад;
|ТОВАР = Документ.РасходнаяНакладная.Товар;
|КОЛИЧЕСТВО = Документ.РасходнаяНакладная.Количество;
(Группировка ТОВАР Упорядочить По ТОВАР.Код;
(Группировка СКЛАД Упорядочить По СКЛАД.Код;
```

```
|Функция КОЛ = Сумма(КОЛИЧЕСТВО);
|//}}ЗАПРОС";
// Если ошибка в запросе, выходим из процедуры.
Если НашЗапрос.Выполнить(ТекстЗапроса) = 0 Тогда
    Возврат;
КонецЕсли;
```

Приведем пример использования запроса. Допустим, в конфигурации «Торговля» версии 9.* понадобилось изменить мелкооптовые цены в зависимости от оптовых цен. Код программы без применения запросов может выглядеть так:

```
Процедура Сформировать()
Тов = СоздатьОбъект("Справочник.Номенклатура");
Цена = СоздатьОбъект("Справочник.Цены");
Если Группа.Выбран() =1 Тогда
    Тов.ИспользоватьРодителя(Группа);
КонецЕсли;
Тов.ВыбратьЭлементы();
Пока Тов.ПолучитьЭлемент() = 1 Цикл
    Если Тов.ЭтоГруппа() = 1 Тогда
        продолжить;
    КонецЕсли;
    состояние(Тов.Наименование) ;
    Цена.ИспользоватьВладельца(Тов);
    Исх = 0;
    Цена.ВыбратьЭлементы();
    Пока Цена.ПолучитьЭлемент() = 1 Цикл
        Если Цена.ТипЦен = ИсхЦ Тогда                j
            Исх = Цена.Цена.Получить(ВыбДата);
            прервать;
        КонецЕсли;
    КонецЦикла;
    Цена.ВыбратьЭлементы();
    Пока Цена.ПолучитьЭлемент() = 1 Цикл
        Если Цена.ТипЦен = ТипЦ Тогда
            Цена.Цена.Установить(ВыбДата, ( (100 + Процент)/
                100)*Исх);
            Цена.Записать();
            прервать;
        КонецЕсли;
    КонецЦикла;
КонецЦикла;
КонецПроцедуры
```

А теперь спроектируем программу с применением запросов.

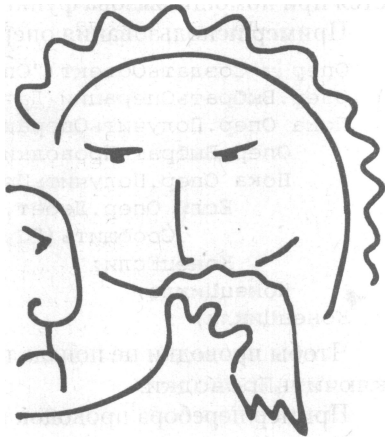
Как и во всяком языке запросов, в начале запроса идет описание переменных, а затем описываются операции с переменными.

Для автоматического формирования кода запросов в «1С» предусмотрен помощник (Конфигуратор • Конструкторы • Запрос). Новичку советую разобраться, как работает конструктор, а не пытаться писать запросы самостоятельно. Писать запросы вручную долго и утомительно.

Приведенный выше пример изменения цен товаров при помощи запроса будет решаться следующим образом:

```
// Процедура генерации запроса Сформировать
Процедура Сформировать()
Перем Запрос, ТекстЗапроса;
// Создание объекта типа Запрос
Цена = СоздатьОбъект("Справочник.Цены");
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса = "//{{ЗАПРОС(Сформировать)
|Период с ВыбДата по ВыбДата;
|Ц = Справочник.Цены.ТекущийЭлемент;
|Тов = Справочник.Цены.Владелец;
|Тип = Справочник.Цены.ТипЦен;
|Цена = Справочник.Цены.Цена;
|Функция Счётчик = Счётчик();
    // Обращаю внимание, что слово Счётчик пишется через ё.
    // Если будет написано через е, запрос выполнен не будет.
(Группировка Тов;
(Группировка Тип;
|Условие(Тов в Группа);
Г//}}ЗАПРОС;
// Если возникла ошибка в запросе, выходим из процедуры.
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
    Возврат;
КонецЕсли;
    // Группировка по товарам
Пока Запрос.Группировка(1) = 1 Цикл
    Если Запрос.Тов.ЭтоГруппа() = 1 Тогда
        продолжить;
    КонецЕсли;
    Исх = 0;
    состояние(Запрос.Тов.Наименование);
        // Группировка по типам цен
    Пока Запрос.Группировка(2) = 1 Цикл
        Если Запрос.Тип = ИсхЦ Тогда
            Исх = Запрос.Цена;
        КонецЕсли;
        Если Запрос.Тип = ТипЦ Тогда
            Цена.НайтиЭлемент(Запрос.Ц);
        КонецЕсли;
    КонецЦикла;
    // В переменной Цена получаем цену, которую нужно
    // пересчитать,
    // в Исх — исходную цену.
    // Для проверки выясняем, все ли выбрано.
    Если Исх = 0 Тогда
        продолжить;
    ИначеЕсли Цена.Выбран() = 0 Тогда
        продолжить;
    КонецЕсли;
    Цена.Цена.Установить(ВыбДата, ( (100 + Процент)/100)*Исх);
    Цена.Записать();
КонецЦикла;
КонецПроцедуры
```

Бухгалтерские итоги



При наличии в системе «1С:Предприятие» компоненты «Бухгалтерия» система предоставляет доступ к механизму, который обеспечивает хранение, динамический пересчет и извлечение бухгалтерских итогов средствами встроенного языка.

Хранение итогов поддерживается системой с детализацией до месяца. В итогах хранятся остатки и обороты по счетам с детализацией по субконто и обороты между счетами (без детализации по субконто).

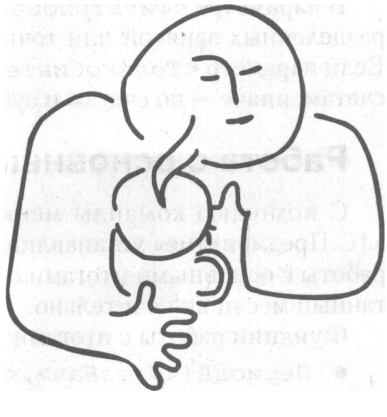
Изменение бухгалтерских итогов может производиться проводками бухгалтерских операций.

Обращение к бухгалтерским итогам выполняется при помощи агрегатного объекта типа БухгалтерскиеИтоги. Объект может работать в трех режимах:

- работа с основными итогами,
- работа с временными итогами,
- работа в режиме запроса.

Объект типа БухгалтерскиеИтоги при создании функцией СоздатьОбъект работает в первом режиме. Переключение в другие режимы производится функциями Рассчитать и ВыполнитьЗапрос. Функции ИспользоватьПланСчетов и ИспользоватьРазделительУчета позволяют назначить План счетов и разделитель учета, по которым будут выдаваться итоги.

Работа с операциями и проводками



Для отражения в бухгалтерском учете информации о движении средств используются объекты типа Операция и Проводка.

Объект 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 формы операции, формы журнала операций и формы журнала проводок.

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

Пример использования операции:

```
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперации(ДатаДок,ДатаДок);
Пока Опер.ПолучитьОперацию() = 1 Цикл
    Опер.ВыбратьПроводки() ;
    Пока Опер.ПолучитьПроводку() = 1 Цикл
        Если Опер.Дебет.Счет = СчетПоКоду("62.1") Тогда
            Сообщить("Субконто " + Опер.Дебет.Клиент);
        КонецЕсли;
^g  КонецЦикла;
КонецЦикла;
```

Чтобы проводки не попадали в бухгалтерские итоги, используется метод Отключить Проводки.

Пример перебора проводок в операциях:

```
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперации(ДатаДок,ДатаДок);
, Пока Опер.ПолучитьОперацию() - 1 Цикл
    Опер.ВыбратьПроводки();
    Пока Опер.ПолучитьПроводку() = 1 Цикл
        Сообщить("Клиент = " + Опер.Дебет.Клиент +" Счет= " +
          Опер.Дебет.Счет);
    КонецЦикла;
КонецЦикла;
```

Работа с временными итогами

Чтобы получить итоги на любую дату, нужно выполнять временный (хранящийся только во время существования переменной типа БухгалтерскиеИтоги) расчет с помощью функции Рассчитать *(<ДатаНач>, <ДатаКон>, «РильтрПоСчетам>, <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>* равен 1, то расчет будет делаться только по счетам, иначе — *по* счетам и субконто.

Работа с основными итогами

С помощью команды меню Управление бухгалтерскими итогами в системе «1С:Предприятие» устанавливается последний рассчитанный период. В режиме работы с основными итогами осуществляется доступ только к итогам по рассчитанный месяц включительно.

Функции работы с итогами:

- ПериодД (*<ДатаНач>, <ДатаКон>)* — установка периода итогов;
- СНД *(<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>,...)* — сальдо начальное дебетовое;

- СНК () — сальдо начальное кредитовое;
- СКД () — сальдо конечное дебетовое;
- СКК () — сальдо конечное кредитовое;
- ДО () — дебетовый оборот по счету за период;
- КО () — кредитовый оборот по счету за период;
- ОБ (*<СчетДеб>, <СчетКред>, <ТипСуммы>, <Валюта>)* — обороты между счетами.

Параметры: *<Счет>* — счет, *<ТипСуммы>:* 1 — сумма, 2 — валютная сумма, 3 — количество, *<Валюта>* — значение типа "Справочник. Валюты", *<Субконто 1>* — значение первого субконто счета, *<Субконто2>* — значение второго субконто счета и т. д.

Для получения остатков и оборотов по счетам, имеющим субсчета, используются аналогичные функции: СНДР (), СКДР (), СКДР (), СККР ().

Работа с бухгалтерскими итогами в режиме запроса

Для получения большего количества итогов (обороты и остатки по разным видам группировок) используются бухгалтерские запросы.

СОВЕТ Чтобы не разбираться в том, что написано ниже, советую научиться пользоваться конструкторами бухгалтерских запросов, которые вы можете найти в Конфигураторе (Конструкторы • Бухгалтерский запрос).

Перед выполнением запроса следует установить фильтры:

- ВключатьСубсчета () — устанавливать режим отбора по субсчетам;
- 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>г <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:

- 1 ("Период") — промежуточные итоги не рассчитываются;
- 2 ("Операция") — промежуточные итоги рассчитываются по операциям;
- 3 ("Проводка") — по проводкам;
- 4("День") — по дням;
- 5 ("Неделя") — по неделям;
- 6 ("Декада") — по декадам;
- 7 ("Месяц")— по месяцам;
- 8 ("Квартал") — по кварталам;
- 9 ("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().`

Функции получения остатков и оборотов такие же, как и при работе с основными итогами.

Примеры использования бухгалтерского запроса

Использование бухгалтерского запроса позволяет выяснить остатки по счетам и получить доступ к операциям.

Первый пример определяет остатки товара на счете 41.1:

```
Ит = СоздатьОбъект("БухгалтерскиеИтоги");
Ит.ИспользоватьСубконто("Номенклатура");
Ит.ВыполнитьЗапрос(Дата1,Дата2,"41.1");
Ит.ВыбратьСчета();
Пока Ит.ПолучитьСчет() = 1 Цикл
    Ит.ВыбратьСубконто(1);
    Пока Ит.ПолучитьСубконто() = 1 Цикл
        Сообщить("Счет " + Ит.Счет.Код + " " + Ит.Субконто(1)
          + Строка(Ит.СКД()));
    КонецЦикла;
КонецЦикла;
```

Второй пример можно использовать для определения документов, которые делали проводки по счету 62.5:

```
Ит.ИспользоватьСубконто(ВидыСубконто.Контрагенты,Клиент, 1);
```

```
Ит.ВыполнитьЗапрос(ВыбНачПериода, ВыбКонПериода, "62.5",,,
  1,"Проводка","С");
Таб.ВывестиСекцию("Шапка");
Таб.Опции(0,0,Таб.ВысотаТаблицы(),0);
Ит.ВыбратьСубконто(ВидыСубконто.Контрагенты);
Пока Ит.ПолучитьСубконто(ВидыСубконто.Контрагенты) = 1 Цикл
    Таб.ВывестиСекцию("Субконто1");
    Ит.ВыбратьПериоды();
    ДО = 0; КО = 0;
    Доку = "";
    Пока Ит.ПолучитьПериод() = 1 Цикл
        Опер = Ит.Операция.ТекущийДокумент();
        Если Доку = "" Тогда
            Доку = Ит.Операция.ТекущийДокумент(); ДО = 0; КО = 0;
        ИначеЕсли Доку о Опер Тогда
// Секцию следует выводить, только когда меняется документ.
            Таб.ВывестиСекцию("Док"); ДО = 0; КО = 0;
            Доку = Ит.Операция.ТекущийДокумент();
        КонецЕсли;
        ДО = ДО + Ит.ДО();
        КО = КО + Ит.КО();
    КонецЦикла;
    Таб.ВывестиСекцию("Док");
КонецЦикла;
```

Схемы переноса информации из одной базы данных в другую

Перенос информации из одной базы данных в другую возможен следующими способами:

- через текстовый файл,
- через файл формата XML,
- при помощи OLE-сервера.

Перенос может отличаться по уровню детализации:

- перенос «документ в документ» — когда структура документа (значения полей, шапка документа и его табличная часть) совпадают;
- перенос сводной информации (когда необязательно переносить всю информацию, а достаточны сводные проводки, например по товарам со ставками НДС 18 и 10%).

Если в процессе переброски документов перенесенные документы не желают проводиться, скорее всего, программа инициализирует не все реквизиты в документе-приемнике, например не заполняет поля фирмы, валюты или курса валюты. Если все поля перенесены корректно, то документы должны проводиться.

Кстати, при переносе расходных накладных следует помнить, что необходимо переносить и справочник Номенклатура, и Контрагенты, а у справочника Номенклатура может быть подчиненный справочник Единицы.

Запись в файл

Самый простой способ записи в файл — вывести текст, который вы хотите сохранить, в поле Окно сообщений, а потом сохранить выведенный текст в текстовом редакторе.

Пример автоматической записи в файл:

```
Текст = СоздатьОбъект("Текст");
Табз.ВыбратьСтроки();
Пока Табз.ПолучитьСтроку() = 1 Цикл
    Текст.ДобавитьСтроку(Табз.Пояснение);
    Текст.ДобавитьСтроку("");
КонецЦикла;
Текст.Записать("__Текст.txt");
```

Как модифицировать отчет, чтобы он записывал данные в файл?

Запись данных во внешний файл может понадобиться, когда требуется сверить остатки за один день из баз данных, сохраненных за разные дни. Для того чтобы написать отчет, который сохранял бы остатки товаров в файле, практически ничего не нужно писать самому. Следует лишь несколько модифицировать стандартный отчет, а именно найти отчет по остаткам товаров и добавить в начало процедуры выдачи остатков следующий код:

```
Список = СоздатьОбъект("СписокЗначений");
Файл = СоздатьОбъект("Текст");
```

Далее надо найти место в коде, где выводится на печать строка с наименованием товара и количеством (обратите внимание на единицы измерения!), и добавить следующие строки:

```
Список.УдалитьВсе();
Список.ДобавитьЗначение(Запрос.Товар.Код);
// или Список.ДобавитьЗначение(Запрос.Товар.ПолныйКод());
Список.ДобавитьЗначение(Запрос.Товар.Наименование);
Список.ДобавитьЗначение(Запрос.КонКол);
Файл.ДобавитьСтроку(Список.ВСтрокуСРазделителями());
```

В конце процедуры следует добавить строку закрытия текстового файла:

```
Файл.Записать("ИмяФайла");
```

Как прочитать текстовый файл?

Приведу пример чтения текстового файла, который содержит код товара и его цену:

```
Список = СоздатьОбъект("СписокЗначений");
Файл = СоздатьОбъект("Текст");
Файл.Открыть(ИмяФайла);
Тов = СоздатьОбъект("Справочник.Товары");
Для стр = 1 по Файл.КоличествоСтрок() Цикл
    Список.УдалитьВсе();
//Метод ИзСтрокиСРазделителями заполняет список значений из строки.
```

```
    // Значения в списке разделяются запятыми.
    Список.ИзСтрокиСРазделителями(Файл.ПолучитьСтроку(стр));
    Тов.НайтиПоКоду(Список.ПолучитьЗначение(1));
    Если Тов.Выбран() = 0 Тогда
        продолжить;
    КонецЕсли;
    Цена = Список.ПолучитьЗначение(2);
КонецЦикла;
```

Некоторые полезные команды типа данных Текст

Для работы с текстами в системе используется тип данных Текст. С его помощью можно записывать строки в текстовые файлы, а также считывать строки из файлов.

Вот некоторые команды типа данных Текст:

- Открыть *(<ИмяФайла>)*.— открыть файл;
- КодоваяСтраница *(<Режим>)* — получить/установить режим кодировки; параметр *<Режим>* принимает значения 0 (Windows-кодировка) и 1 (DOS-кодировка);
- Записать *(<ИмяФайла>)* — записать текст в файл;
- КоличествоСтрок () — возвращает количество строк в тексте;
- ПолучитьСтроку *(<НомерСтроки>)* — получить строку текста по номеру;
- ДобавитьСтроку *(<Строка>)* — добавить строку в конец текста;
- Очистить () — удалить все строки текста;
- Показать *{<Заголовок>, <ИмяФайла>)* — открыть окно редактирования текста.

Работа с файлами в формате DBF

Для работы с базами данных используется объект типа XBase. Объект XBase предоставляет монопольный доступ к таблицам DBF (то есть файл может быть доступен только одной программе).

Доступ к полям базы данных осуществляется через точку с именем поля.

Пример работы с файлом в формате DBF:

```
БДФ = СоздатьОбъект("XBase");
БДФ.ОткрытьФайл("data.dbf");
Если БДФ.Открыта() = 1 Тогда
    БДФ.Первая();
    Пока 1 = 1 Цикл
        Сообщить("Запись " + Строка(БДФ.CODE) + " " + БДФ.Name);
        Если БДФ.Следующая() =0 Тогда
            Прервать;
        КонецЕсли;
```

```
    КонецЦикла;
ДБФ.ЗакрытьФайл();
КонецЕсли;
```

Кстати, может быть, вы не знаете, что файлы в формате DBF можно просматривать в программе Microsoft Excel.

Работа с файловой системой

Для работы с файловой системой используется тип данных ФС.
Вот некоторые функции этого типа данных:

- ВыбратьФайл () — открывает диалог выбора файла,
- ВыбратьКаталог () — открывает диалог выбора каталога,
- СуществуетФайл () — проверяет, существует ли файл с указанным именем,
- КопироватьФайл (),
- УдалитьФайл (),
- ПереименоватьФайл ().

Пример выбора пути к файлу:

```
Процедура ВыборФайла()
    ИмяПути = Константа.ПутьКФайлам;
    Если ФС.ВыбратьФайл(0,ИмяФайла,ИмяПути,"Выберите файл
       загрузки",
          "Все файлы(*.*)|*.*",,) = 1 Тогда
          ИмяФайла = ИмяПути + ИмяФайла;
    КонецЕсли;
КонецПроцедуры
```

Перенос данных при помощи XML-технологии

XML — это язык разметки. С помощью XML создаются древовидная структура хранения данных и правила построения дерева, что уменьшает вероятность ошибок, связанных с доступом к данным.

Смотрите в Интернете статьи, посвященные XML, по следующим ссылкам:
www.bolero.ru/cgi-bin/dsc.cgi?540449&partner=politen,
www.raleigh.ru/XML/2001/10points.php,
chin.dataltd.ru/index2.php?id=1c_xml1,
chin.dataltd.ru/index2.php?id=1c_xml2,
chin.dataltd.ru/index2.php?id=1c_xml3.

Работа с внешними программами

Следующий пример показывает, как можно записывать данные в таблицу Excel:

```
Окно = СоздатьОбъект("Excel.Application");
Окно.Visible = 1;
Окно.Caption = "Отчет";
```

```
Окно.Workbooks.Add();
Для п = 1 По 10 Цикл
    Ячейка = Окно.Cells(п, 1);
    Ячейка.Value = п;
КонецЦикла;
```

Также вполне возможно, что вам будут полезны следующие команды:

```
BExcel.Workbooks.Open(ИмяФайла);
Окно = BExcel.Worksheets (1);
кол = Окно.Rows.Count;
А = Окно.Range("G5")-Value;
Б = Окно.Cells(4,8).Value;
```

Работа с «1С» из «1С» (применение OLE-технологий)

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:

   ```
   OLE = СоздатьОбъект("V77.Application");
   ```

2. Выполняется инициализация системы «1С:Предприятие» методом Initialize:

   ```
   Рез = OLE.Initialize(OLE.RMTrade,КомандняяСтрока,"");
      Если Рез = 0 тогда
          Сообщить("Не могу открыть базу");
          Возврат;
      КонецЕсли;
   ```

   КомандняяСтрока — это переменная, в которой указывается командная строка запуска «1С:Предприятие». Например:

   ```
   КомандняяСтрока = "/DServer\база /^Программист";
   ```

   Это означает, что следует запустить «1С:Предприятие» из каталога \\Server\ база с именем пользователя Программист.

3. Вызываются атрибуты и методы системы «1С:Предприятие» как OLE Automation-сервера.

Вы можете использовать приведенный ниже пример, чтобы получить доступ к справочнику Контрагенты, который находится в другой базе данных. Если программа не работает, возможно, требуется реиндексировать базу данных, которую вы вызываете, или попробовать зайти в открываемую базу данных под другим именем.

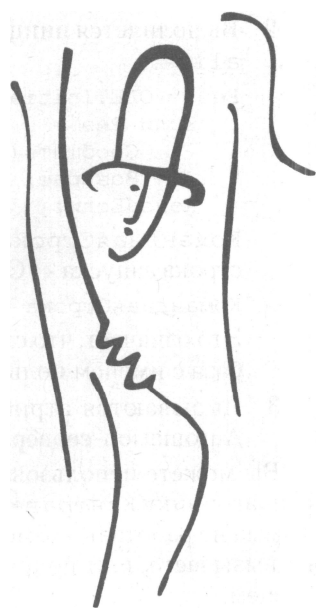
```
Процедура  Сформировать()
    OLE = СоздатьОбъект("V77.Application") ;
    Рез = OLE.Initialize(OLE.RMTrade,"","NO_SPLASH_SHOW");
    Если Рез = 0 Тогда
        Сообщить("Не могу открыть базу");
        Возврат;
    КонецЕсли;
```

```
    Кли1  =  OLE.CreateObject("Справочник.Контрагенты");
    ЮрЛ1  =  OLE.CreateObject("Справочник.ЮрЛица");
    Физ1  =  OLE.CreateObject("Справочник.ФизЛица");

    Кли1.ВыбратьЭлементы();
    Пока Кли1.ПолучитьЭлемент()  =  1  Цикл
        Если Кли1.ЭтоГруппа()  =  1  Тогда
            Продолжить;
        КонецЕсли;
        Если Кли1.ЮрФизЛицо.Вид()  =  "ЮрЛица"  Тогда
            ЮрЛ1.НайтиПоКоду(Кли1.ЮрФизЛицо.Код);
            ТабзИмяТорг  =  (лев(сокрлп(ЮрЛ1.Наименование),30)),
        Иначе
            Физ1.НайтиПоКоду(Кли1.ЮрФизЛицо.Код);
            ТабзИмяТорг  =  (лев(сокрлп(Физ1.Наименование),30))
        КонецЕсли;
        Сообщить(ТабзИмяТорг);
    КонецЦикла;
КонецПроцедуры
```

Периодический тип данных

Периодическая величина — это такая величина, которая имеет различное значение в разные дни (то есть имеет значение, связанное с датой). При изменении значения периодического реквизита старое значение сохраняется, при этом новое начинает действовать с указанной даты, а старое — до указанной даты. К периодическим величинам, например, относится фамилия женщины (в девичестве она имеет одно значение, а после замужества — другое).



```
Сообщить("Сотрудник  " +
Сотрудник.Фамилия.Получить(ДатаДок));
```

Для работы с периодическими величинами возможна конструкция ИспользоватьДату():

```
Тов =
СоздатьОбъект("Справочник.Номенклатура");
Тов.ИспользоватьДату(ДатаДок);
Тов.ВыбратьЭлементы();
Пока Тов.ПолучитьЭлемент()  =  1  Цикл
    Сообщить("Ставка НДС  " +
Тов.СтавкаНДС);
КонецЦикла;
```

Удаление элементов объектов

Для удаления объекта (элемента справочника, документа) используется метод Удалить *(<Режим>),* где параметр *<Режим>* принимает значения 0 (пометка

на удаление) и 1 (непосредственное удаление, используется по умолчанию). Использовать режим непосредственного удаления объектов не рекомендуется из-за возможного нарушения ссылочной целостности базы данных.

Пример программы удаления документов:

```
Процедура Удалялки()
Док = СоздатьОбъект("Документ.РасходнаяНакладная");
Док.ВыбратьДокументы();
Пока Док.ПолучитьДокумент() = 1 Цикл
    Док.Удалить();
КонецЦикла;
КонецПроцедуры
```

Чтобы удалить помеченные на удаление объекты через систему «1 (^Предприятие» 7.7, нужно зайти в программу в монопольном режиме и в меню Операции выбрать пункт Удаление помеченных объектов.

Иногда удаление может быть более сложным, например, когда необходимо удалить группу товаров. В этом случае нужно учитывать подчиненные справочники и следить за тем, чтобы не удалить нужные элементы:

```
Процедура Удалялки()
Цена = СоздатьОбъект("Справочник.Цена");
Спр = СоздатьОбъект("Справочник.Номенклатура");
Спр.ИспользоватьРодителя(Группа);
Спр.ВыбратьЭлементы();
Пока Спр.ПолучитьЭлемент() = 1 Цикл
    Если Спр.ЭтоГруппа() = 1 Тогда
        продолжить;
    КонецЕсли;
    Цена.ИспользоватьВладельца(Спр.ТекущийЭлемент());
    Цена.ВыбратьЭлементы();
    Пока Цена.ПолучитьЭлемент(.) = 1 Цикл
        Цена.Удалить(0);
    КонецЦикла;
    Спр.Удалить(0);
КонецЦикла;
КонецПроцедуры
```

Работа с транзакциями

Система автоматически использует транзакции при проведении документов. Когда в одной базе данных одновременно работают несколько пользователей, то при записи документа возможна попытка «одновременного» доступа двух документов к базе данных. Поэтому система выполняет проведение документа в режиме транзакции, то есть база данных на время проведения документа становится доступной для записи только одного документа.

Транзакции выполняются системой автоматически и могут быть заданы пользователем. Транзакции, задаваемые пользователем, применяются для выполнения длительных и критических для функционирования системы операций с возможностью отката.

В программе можно начать и зафиксировать транзакцию с помощью следующих функций:

```
НачатьТранзакцию();

[ОтменитьТранзакциюО;]

ЗафиксироватьТранзакцию();
```

Отмена транзакции отменяет все изменения, внесенные в базу данных с момента начала транзакции.

Регистры

Регистры — это таблицы для накопления оперативных данных и получения сводной информации. Данные в регистры добавляются только при проведении документов. Сведения из регистров используются для формирования отчетов.

Виды регистров

В системе «1С:Предприятие» возможно использование регистров двух типов: *регистры остатков* и *регистры оборотов*. Разница между ними понятна из их названия и заключается в характере хранимой информации: в регистрах остатков всегда хранится информация о конечном состоянии средств, а в регистрах оборотов, образно выражаясь, — как это состояние было достигнуто. Если нужно быстро получать остаток на текущий момент, тогда нужно сделать регистр остатков. Если нужно быстро получать приход или расход за период, тогда нужно сделать оборотный регистр.

Что лучше: бухгалтерские итоги или регистры?

Известный принцип гласит: выигрыш во времени достигается усложнением программирования. Оперативный (регистровый) учет, с точки зрения конечного пользователя, — структура менее гибкая по сравнению с бухгалтерскими итогами, однако она работает быстрее.

При проектировании регистров программист имеет возможность спуститься почти до уровня базы данных. Программист определяет структуру полей регистра и тем самым определяет тип оптимизации — по скорости или по полноте учитываемой информации.

В программах, которые реализуют торговый учет, реализовано два регистра, по принципу скорости и полноты учета:

- быстрый регистр — регистр остатков (5 полей),
- регистр с полной аналитикой — регистр партий (17 полей).

Регистр остатков используют для определения остатков товаров, находящихся на складе, а регистр партий используется для определения сумм себестоимости товара.

Скорость, с которой работает регистр ОстаткиТовара, нельзя получить при помощи регистра Партии или при использовании возможностей бухгалтерского учета.

К недостаткам регистров можно отнести сложность в организации аналитических отчетов. Например, в случае реорганизации регистра (введения или удаления какого-либо измерения) необходимо переработать все программы, использовавшие этот регистр. В то же время после добавления нового счета в бухгалтерском учете его можно сразу же использовать и получать аналитические отчеты при помощи стандартных отчетов.

Бухгалтерские итоги — это, в противоположность регистрам, более гибкая с точки зрения конечного пользователя система. Она реализована при помощи Плана счетов и бухгалтерских проводок и понятна не только программисту, но и бухгалтеру.

Пользователь может сам подстраивать систему бухгалтерских итогов при минимальных вмешательствах со стороны программиста. К недостаткам этого метода учета итогов можно отнести более медленную скорость работы по сравнению с использованием регистров.

Регистры остатков

Регистр остатков позволяет получать следующие виды информации в разрезе определенных измерений:

- начальный остаток,
- приход,
- расход,
- конечный остаток.

Рассмотрим в качестве примера отслеживание взаиморасчетов организации с покупателями товаров, которые она производит или продает. Для того чтобы оперативно получать информацию о взаимной задолженности предприятия и покупателя, потребуется регистр Взаиморасчеты, в котором для каждого покупателя будет храниться сумма задолженности. При совершении хозяйственной операции состояние регистра будет соответствующим образом изменяться, каждый раз отражая текущее состояние взаиморасчетов. Регистр Взаиморасчеты — это регистр остатков.

Например, чтобы рассчитать остатки регистров, которые были до проведения документа Док, следует написать следующий код:

```
ВремРегистры = СоздатьОбъект("Регистры");
Рег = ВремРегистры.ОстаткиТоваров;
Рег.ВременныйРасчет();
ВремРегистры.РассчитатьРегистрыНа(Док.ТекущийДокумент());
ОстатокТовара =
Рег.Остаток(ВыбФирма,Товар,Склад,"ОстатокТовара");
```

С помощью такого кода вы можете в любой момент получить остаток товара на складе.

Для того чтобы определить остатки регистров, которые стали после проведения документа, следует использовать метод РассчитатьРегистрыПо.

Оборотный регистр

Оборотный регистр, в отличие от регистра остатков, выдает итоги за период, определенный для этого регистра. При помощи регистра остатков тоже можно получить итог за период, но для этого нужно будет сконструировать запрос.

Например, для решения описанной ниже задачи проще использовать оборотный регистр, а не регистр остатков.

Требуется предоставлять клиенту скидку в размере 5 %, если оборот клиента за предыдущие 3 месяца превышает 100 000 р.

Сконструируем следующий оборотный регистр с периодичностью месяц:

Измерения:

- Клиент (тип значения — справочник Контрагенты),
- Док (тип значения — Документ),

Ресурсы:

- Сумма (тип значения — Число).

Для создания регистра следует зайти в Конфигуратор, выделить мышью узел Регистры, нажать правую кнопку мыши и выбрать в контекстном меню пункт Новый Регистр. Тип регистра (оборотный, остаточный) определяется полем Тип регистра.

Приведенная ниже программа рассчитывает обороты клиента за прошедшие три месяца

```
// Определение месяца и года
Функция м(Дат, г)
    Д = Док.ДатаДок;
    мес = ДатаМесяц(Д) - 1;
    г = ДатаГод(Д);
    Если мес < 1 Тогда
        г = г - 1;
        мес = 12;
    КонецЕсли;
возврат мес;
КонецФункции
Per = СоздатьОбъект("Регистр.ОборотыКлиента");
Сум = 0;
г = 0;
// Определяем итоги за прошлый месяц.
мес = м(Док.ДатаДок - 1,г);
Per.ИспользоватьПериод(г,мес);
Сум = Сум + Per.СводныйИтог(Кли,,"Сумма");
// Определяем итоги, которые были два месяца назад.
мес = м(Док.ДатаДок - 2,г);
Per.ИспользоватьПериод(г,мес);
Сум = Сум + Per.СводныйИтог(Кли,,"Сумма");
// Определяем итоги, которые были три месяца назад,
мес = м(Док.ДатаДок - 3,г);
Per.ИспользоватьПериод(г,мес);
Сум = Сум + Per.СводныйИтог(Кли,,"Сумма");
```

Использование оборотного регистра позволяет быстро получать суммы некоторых движений за определенный период. Такая необходимость может возникнуть, например, если надо очень быстро получить отчет об объемах продаж определенному клиенту за некоторый период времени.

За увеличение скорости обработки, полученное введением оборотных регистров, приходится платить использованием дополнительных дисковых ресурсов.

С точки зрения реализации движений, оборотный регистр аналогичен регистру остатков, а в части итогов объем хранимой информации*зависит от выбранного периода оборотного регистра. В периодах с «дня» до «месяца» объем информации будет пропорционален количеству данных периодов в месяце при постоянном наборе значений. Заметим, что периоды «квартал» и «год» по задействованным ресурсам должны превосходить период «месяц».

Обычно нет необходимости мгновенно получать оборотные итоги, поэтому требуемая информация может быть получена при помощи запроса. Наиболее веским аргументом для создания оборотного регистра может служить необходимость получения оборота при проведении документа или необходимость четко регламентировать отражение некоторых фактов хозяйственной деятельности при проведении документов различных видов.

Многие конфигурации, использующие регистры, построены только на регистрах остатков и не используют оборотных регистров.

Заметим, что при работе с базами данных в формате SQL роль оборотных регистров с точки зрения оптимизации построения отчетов существенно снижается.

Измерения и ресурсы

Основная проблема при проектировании регистров — это определение его структуры. Структура регистра должна быть такова, чтобы можно было извлекать из него нужную информацию без утомительной обработки.

Измерения регистра — это то, в каких разрезах требуется хранение информации. Ресурсы регистра — это количественные или суммовые данные, которые хранятся в регистре.

Предположим, что регистр ОстаткиТоваров имеет следующую структуру:

Измерения:

- Товар,
- Склад.

Ресурсы:

- Количество,
- Стоимость.

В идеологии системы «1 (^Предприятие» регистр такого вида представляет собой прямоугольную систему координат, на одной оси которой находятся склады, на другой — товары, а на пересечении конкретного склада и конкретного товара находятся цифры количества товара и его стоимости.

С помощью методов встроенного языка мы можем легко узнать:

- остаток конкретного товара на конкретном складе,
- остаток конкретного товара на всех складах,
- стоимость всех товаров на конкретном складе.

Конструирование регистра

Регистр — это средство для накопления информации. Регистры можно менять только при проведении документов. Регистр содержит измерения, ресурсы и реквизиты.

Состав ресурсов регистра определяется тем, в каких запросах он будет использоваться. Например, если регистр будет использоваться в запросе: «Сколько на заданном складе находится единиц заданного товара по заданной цене», то регистр можно спроектировать следующим образом:

Измерения:

- первое — Склад,
- второе — Товар,
- третье — Цена.

Реквизит:

- Количество.

Для получения сводных остатков используется метод СводныйОстаток. Функции получения остатков или оборотов из регистра предполагают обязательное заполнение полей измерений слева направо. То есть для приведенной выше схемы, если вы не задали Товар, то не можете задать Склад.

То есть такой запрос сформировать нельзя:

```
Рег.СводныйОстаток(Склад,,Цена,"Количество") ,
```

а такие запросы составлены верно:

```
Рег.СводныйОстаток(Склад,Товар,,"Количество"),
Рег.СводныйОстаток(Склад,,,"Количество"),
```

При конструировании регистра важно правильно определить порядок измерений. Например, если в будущем должен формироваться такой запрос: «Сколько находится товара у фирмы Б на складе А, товар не важен», то структура регистра будет следующей:

- первое измерение — Фирма,
- второе измерение — Склад,
- третье измерение — Товар.

А для того чтобы формировался такой запрос: «Сколько находится товара А у фирмы Б, склад не важен», структура регистра будет другой:

- первое измерение — Фирма,
- второе измерение — Товар,
- третье измерение — Склад.

Нужно ли предусматривать измерение Фирма при проектировании регистров?

Иногда клиенты говорят: «У нас только одна фирма, и никакую другую мы создавать не намерены». Не верьте сказанному: скорее всего, через некоторое время окажется, что на самом деле работают одновременно две фирмы, например ЧП, работающее на вмененном налоге, и предприятие — плательщик НДС. Так устро-

ено законодательство, что с целью минимизации налоговых платежей под крышей одного предприятия собирается несколько индивидуальных предпринимателей. Поэтому следует заранее предусмотреть возможность работы с несколькими фирмами.

Реквизиты регистра

Реквизиты используются для организации фильтров по движениям регистров. Отличие реквизитов от измерений состоит в том, что конечный (начальный) остаток по реквизиту получить нельзя. Остатки можно получать только по измерению.

Приведем пример использования реквизита. Допустим, требуется составить отчет по движению товаров на складах в котором при выбранном складе, обороты, составленные перемещениями (внутренними документами фирмы), не должны учитываться.

Проанализируйте, чем различаются формирования движений регистра ОстаткиТМЦ у документов «перемещение» и «реализация» (конфигурации 9.*). Одно из различий движений регистров заключается в том, что перемещение записывает в реквизит Внутреннее единицу, а приходная накладная и реализация — ноль.

Чтобы отделить внутренние перемещения от внешних, достаточно добавить в запрос следующие строки:

```
|Функция КоличествоПриходВнутр = Приход(Количество) когда
(Внутреннее = 1);
|Функция КоличествоРасходВнутр = Расход(Количество) когда
(Внутреннее = 1);
```

ПРИМЕЧАНИЕ Смотрите отчет Ведомость по остаткам ТМЦ в стандартной конфигурации «Торговли» 9.*.

Запись движения регистров

Вот так может выглядеть процедура записи движений регистров:

```
Процедура ДвижениеРегистраДолги(Док) Экспорт
    Рег = Док.Регистр.Долги;
    Вид = Док.Вид();
    Если Вид = "ВводОстатковКредита" Тогда
        Рег.Клиент = Док.Клиент;
        Рег.Агент = Док.Агент;
        Рег.Рн = Док.ТекущийДокумент();
        Рег.Сумма = Док.Сумма;
        Рег.ДвижениеПриходВыполнить();
    ИначеЕсли Вид = "РасходнаяНакладная" Тогда
        Рег.Клиент = Док.Клиент;
        Рег.Агент = Док.Агент;
        Рег.Рн = Док.ТекущийДокумент();
        Рег.Сумма = Док.Итог("Сумма") + Док.Итог("СуммаНП");
        Рег.ДвижениеПриходВыполнить();
    ИначеЕсли Вид = "ПриходныйОрдерТБ" Тогда
        Рег.Клиент = Док.Клиент;
        Рег.Агент = Док.Агент;
```

```
        Рег.Рн = Док.ДокументОснование;
        Рег.Сумма = Док.Сумма;
        Рег.ДвижениеРасходВыполнить();
    Иначе
        Сообщить("Данный документ не двигает долги");
    КонецЕсли;
КонецПроцедуры
```

В примере используется регистр Долги. Этот регистр содержит измерения: Клиент, Агент, РасходнаяНакладная — и ресурс Сумма. Регистр предназначен для учета долгов клиентов. Измерение Рн используется для учета погашения конкретного документа.

Данная процедура должна быть расположена в глобальном модуле (об этом свидетельствует строка

```
Процедура ДвижениеРегистраДолги(Док) Экспорт.
```

Процедура может вызываться из документов любого вида при их проведении.

Проблемы с регистрами, возникающие у новичков

Перед тем как программировать, попробуйте представить, используете ли вы при постановке задачи следующие термины: *начальный остаток, приход, расход, конечный остаток*. Если используете, то вам следует искать бухгалтерский счет или регистр, который дал бы нужную вам информацию.

Проблема программистов-новичков заключается в том, что они не понимают, какие задачи следует решать с использованием бухгалтерских итогов (регистров), а какие можно реализовать перебором документов. Так, например, первый отчет по определению просроченной задолженности новичок будет писать без использования регистров. Такая программа очень быстро раздуется и впоследствии будет требовать постоянных доработок. Любое усовершенствование программы будет приводить к ее разрушению. Если программа, которую вы составили, превышает 200 строк, то это может служить поводом задуматься, не пошли ли вы не тем путем.

Попробуем на примерах определить, когда нужно использовать регистры, а когда можно обойтись без них.

Пример 1. Следует спроектировать реестр расходных накладных.

Реестр не требует определения остатков, следовательно, для реализации реестра использовать бухгалтерские счета необязательно.

Пример 2. Требуется определить, сколько кег (возвратной тары) находится у клиента.

Задание дано на непонятном для программиста языке. Следует переформулировать задание так, чтобы оно содержало термины *остаток, приход* или *расход,* например так: определить *конечный остаток* кег у заданного клиента.

Поскольку теперь в задании используется термин «конечный остаток», понятно, что следует использовать регистр (бухгалтерский счет), ответственный за хранение кег в разрезе определенного клиента:

СчетКеги {Кеги} {Клиенты}.

Временный расчет регистров

Временный расчет регистров требуется, если нужно получить итоги на определенную дату. Если временный расчет не используется, то итоги регистров выдаются на последний проведенный документ *(точку актуальности)*.

1. Метод РассчитатьРегистрыНа*(<ГраницаРасчета>,<ГрафаОтбора>)* — рассчитать все регистры с установленным флагом временного расчета на начало события (на начало даты или на момент до проведения документа).
2. Метод РассчитатьРегистрыПо*(<ГраницаРасчета>,<ГрафаОтбора>)* — рассчитать все регистры с установленным флагом временного расчета на конец события (на конец даты или на момент после проведения документа).

Пример иллюстрирует работу временного расчета регистров:

```
РеР = СоздатьОбъект("Регистр.Резервы");
РеР.ВременныйРасчет();
РеР.УстановитьФильтр(Док.Склад,,,);
РассчитатьРегистрыНа(Док.ТекущийДокумент());

Кол =
РеР.СводныйОстаток(Док.Склад,Док.Товар,Док.Партия,,"Резерв");
```

Что такое точка актуальности?

Точка актуальности (ТА) — это запись регистра остатков, которая показывает актуальный (текущий) остаток. Обычно ТА стоит на последнем документе в базе данных. Документ, на котором стоит ТА, подчеркнут красной полосой в журнале документов.

Если требуется определить остатки на ТА, то их можно взять из последней записи. Если же требуется определить остатки не на ТА, то для этого следует определить разницу между ТА и требуемым временем. Для расчета итогов не на ТА используется временный расчет регистров.

Использование метода УстановитьЗначениеФильтра

Метод УстановитьЗначениеФильтра используется для ускорения расчетов. Сравните следующие примеры. Первый работает на 30 % быстрее и выглядит более компактным.

Пример 1:

```
РегТовары = СоздатьОбъект("Регистр.ОстаткиТоваров");
РегТовары.УстановитьЗначениеФильтра("Склад",ТекСклад);
РегТовары.ВыбратьДвижения(ДатаНач,  ДатаКон)/
Пока РегТовары.ПолучитьДвижение() = 1 Цикл
```

```
    Сообщить("Склад = " + РегТовары.Склад + " Остаток = "
     +РегТовары.ОстатокТовара);
    // ...
КонецЦикла;
```

Пример 2:

```
РегТовары = СоздатьОбъект("Регистр.ОстаткиТоваров");
РегТовары.ВыбратьДвижения(ДатаНач, ДатаКон);
Пока РегТовары.ПолучитьДвижение() = 1 Цикл
    Если РегТовары.Склад <> ТекСклад Тогда
        Продолжить;
    КонецЕсли;
    Сообщить("Склад = " + РегТовары.Склад + " Остаток = " +
      РегТовары.ОстатокТовара);
    // ...
КонецЦикла;
```

Использование методов Остаток и СводныйОстаток

Методы Остаток и СводныйОстаток определяют остаток ресурса регистра.

```
Рег = СоздатьОбъект("Регистр.Товары");
Кол1 = Рег.Остаток(ВыбФирма,ВыбСклад,ВыбТовар,"Количество");
Кол2 = РегТовары.СводныйОстаток(ВыбФирма,ВыбСклад,,"Количество");
Кол3 = РегТовары.СводныйОстаток(ВыбФирма,,,"Количество");
```

- Кол1 содержит остаток (количество) товара на выбранной фирме, выбранном складе, по выбранному товару.
- Кол2 содержит остаток (количество) товара на выбранной фирме, выбранном складе, по *всем* товарам.
- Кол3 содержит остаток (количество) товара на выбранной фирме, на *всех* складах, по *всем* товарам.

При использовании метода Остаток должны указываться все измерения регистра (ВыбФирма, ВыбСклад, ВыбТовар, "Количество"), а в случае использования метода СводныйОстаток может быть указана часть измерений:

```
ОстКол = СводныйОстаток(ВыбФирма,ВыбСклад,,"Количество");
```

Измерения должны заполняться в том порядке, как они указаны в Конфигураторе. Например, для приведенного выше примера структура регистра следующая:

```
1)Фирма,
2)Склад,
3)Товар,
```

то есть следующие записи некорректны:

```
Кол4 = Рег.СводныйОстаток (ВыбФирма,,ВыбТовар,"Количество");
Кол5 = Рег.СводныйОстаток
(ВыбФирма,ВыбСклад,ВыбТовар,"Количество");
Кол6 = Рег.СводныйОстаток (,ВыбСклад,ВыбТовар,"Количество");
```

Использование методов регистров

Для учета себестоимости в конфигурации «Торговля» предусмотрен регистр партий товаров. Для расчета цены себестоимости списываемого товара рассчиты-

вается остаток товара в количестве и сумме, а затем суммовой остаток товара делится на количественный остаток,

Использования регистра в режиме итогов

Приведенная ниже процедура находится в глобальном модуле конфигурации и вызывается из модуля документа. Процедура определяет сумму себестоимости товара, которую следует списать при проведении расходной накладной.

```
Процедура ПартииТоваров(Док)  Экспорт
Рег = Док.Регистр.Партии;
Если (Док.Вид() = "РасходнаяНакладная") Тогда
    Рег_ = СоздатьОбъект("Регистр.Партии");
    // Если точка актуальности не на документе,
    // то рассчитываются временные итоги регистров.
    Если Док.СравнитьТАО = -1 Тогда
        Рег_.ВременныйРасчет();
        Рег_.УстановитьФильтр(Док.Фирма,);
        РассчитатьРегистрыНа(Док.ТекущийДокумент());
    Иначе
        Рег_.УстановитьФильтр(Док.Фирма,);
    КонецЕсли;
    Док.ВыбратьСтроки();
    Пока Док.ПолучитьСтроку() - .1 Цикл
        // Рассчитываются остатки регистра (количество и сумма)
        // для определения себестоимости товара.
        остК = Рег_.Остаток(Док.Фирма,Док.Товар,"Количество");
        остР = Рег_.Остаток(Док.Фирма,Док.Товар,"Сумма");
        // Если количество товара равно нулю, то цена равна нулю.
        Цена = ?(остК = 0,0,остР/остК);
        // Формируются записи в регистр.
        Рег.Фирма = Док.Фирма;
        Рег.Товар = Док.Товар;
        Рег.Количество = Док.Количество*Док.Коэффициент;
        Рег.Сумма = Док.Количество*Док.Коэффициент*Цена;
        Рег.ДвижениеРасходВыполнить();
    КонецЦикла;
КонецЕсли;
КонецПроцедуры
```

Прошу обратить внимание, что в данном примере используется регистр партии в двух объектах.

Рег — это регистр контекста документа (над ним производится действие ДвижениеРасходВыполнить), а Рег_ — это регистр, который используется для расчетов итогов (остатка количественного и суммового).

Если вы попробуете перепутать объекты, например выполнить команду Рег__. ДвижениеРасходВыполнить вместо Рег .ДвижениеРасходВыполнить, то компилятор выдаст ошибку.

Обращение к регистрам через использование запросов

В запросах к регистрам применяются функции НачОст, КонОст, Приход, Расход. В запросах к оборотным регистрам обязательно указывается Период.

Ниже приведен пример запроса для оборотного регистра Доходы. Запрос рассчитывает оборот товара, отгруженного определенному клиенту.

```
ТекстЗапроса = "
|Период С ДатаНач По ДатаКон/
|Товар = Регистр.Доходы.Товар/
|Клиент = Регистр.Доходы.Клиент/
|Доход = Регистр.Доходы.Доход;
|Условие (Товар = ТекТовар);
(Группировка Клиент/
|Функция ПриходПоКлиенту = Приход(Доход)/
Г/
```

Использование метода ВыбратьДвижения

Метод ВыбратьДвижения инициирует выбор движений регистра за определенный период. Пример показывает, как можно просмотреть движение товаров за период с ДатаНач по ДатаКон:

```
РегТовары = СоздатьОбъект("Регистр.Товары");
РегТовары.ВыбратьДвижения(ДатаНач, ДатаКон);
Пока РегТовары.ПолучитьДвижение() = 1 Цикл
    Сообщить("Товар " + РегТовары.Товар + " Стоимость " +
РегТовары.Стоимость)/
КонецЦикла/
```

Использование метода ВыбратьДвиженияДокумента

Метод ВыбратьДвиженияДокумента инициирует движения регистра по заданному документу. Приведенный ниже пример определяет себестоимость товара, записанного в документе Докум:

```
РегТов = СоздатьОбъект("Регистр.ПартииТоваров")/
РегТов.ВыбратьДвиженияДокумента(Докум.ТекущийДокумент())/
Пока РегТов.ПолучитьДвижение() > 0 Цикл
    Стоим = Стоим + РегТов.Стоимость/
КонецЦикла/
```

Использование метода ПолучитьИтог

Методы ВыбратьИтог и ПолучитьИтог позволяют сделать выборку итогов по регистру. Программа показывает остатки товара:

```
РегТовары = СоздатьОбъект("Регистр.Товары");
РегТовары.ВыбратьИтоги();
Пока РегТовары.ПолучитьИтог() = 1 Цикл
    Сообщить("Товар " + РегТовары.Товар +'
    " на складе " + РегТовары.Склад +
    " кол-во: " + РегТовары.Количество +
    " стоимость: " + РегТовары.Стоимость/
КонецЦикла/
```

Как искать примеры программирования?

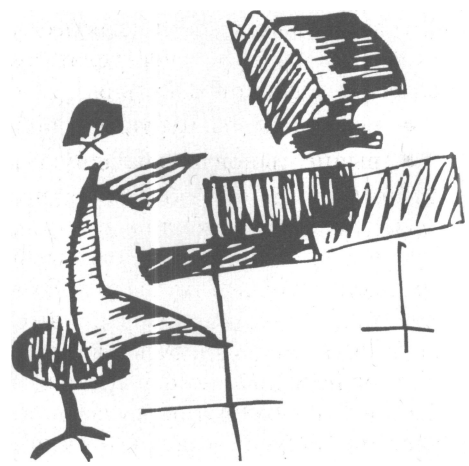
Когда я учился программировать, я прежде всего искал примеры, из которых можно было бы взять нужный мне код. Думаю, что для новичка программирование по шаблонам — наилучший способ программирования и обучения.

Самым большим источником шаблонов является сама «1С». Если посмотреть глобальный модуль, то можно найти огромное количество всевозможных примеров и способов программирования. Единственная трудность заключается в том, как в таком многообразии найти нужный код. Здесь вам может помочь только «гуру». К сожалению, ни один учебник по программированию не способен научить программировать, он может только подсказать, как решить ту или иную задачу. Но в учебнике можно рассмотреть только инструменты, а вот научить их применять может только человек. Поэтому ищите своего «гуру» и учитесь у него, что и когда применять и что в какой последовательности изучать.

Глава 5
Администрирование 1С-приложений

Эта глава посвящена вопросам, связанным с «1С», которые можно отнести к программированию с большой натяжкой, например, порядок обновления базы данных, контроль работы задним числом, внесение изменений в базу данных, «лечение» базы данных, проблемы, которые могут возникнуть при работе с распределенными базами данных.

Почему нельзя работать задним числом?



Работа задним числом может привести к образованию отрицательных остатков на складах.

Допустим, что в 11:03 один менеджер переместил товар на другой склад. Другой менеджер в 11:05 задним числом (в 11:01) выписал тот же товар под остаток другим документом. Таким образом, получается, что в 11:05 на складе будет отрицательный остаток товара, что в принципе невозможно.

Кроме этого, выписанный и проведенный документ потом может быть изменен, удален или перепроведен. Большинство клиентов клянутся в том, что у них все менеджеры умные, все таймеры синхронизированы и то, что описано выше, произойти не может. Не верьте этим словам!

Оптовое предприятие не может работать без переделок и изменений задним числом. В оптовой деятельности процесс выписки товара, отгрузки и оплаты товара растянут во времени и зависит от многих сотрудников обеих фирм, поэтому и возникают ошибки.*

А вот в розничной торговле изменений задним числом практически не бывает, поскольку оплата, отгрузка товара и его проверка по качеству происходят одновременно, и документы оплаты и отгрузки могут быть объединены.

Рассмотрим пример, который объясняет, почему возможны правки задним числом при оптовой продаже.

Клиент выписал товар и собирается оплачивать товар банковским перечислением. В этом случае возможны следующие ошибки и правки:

- ошибки при погрузке,
- ошибки при доставке,

- возврат при отказе клиента принять товар.

Самое простое решение возникших проблем — изменить первичный документ (задним числом) в соответствии с вышеперечисленными ошибками.

Если документ был изменен задним числом, то все документы, которые связаны с этим документом и расположены позже изменяемого документа, *должны быть перепроведены*. Например, должны быть перепроведены все документы, в которых есть тот же товар, что и в измененном документе, иначе себестоимость будет считаться неверно. В «Торговле» предусматривается специальная процедура восстановления последовательности документов (Операции • Проведение документов • Последовательности).

У процедуры восстановления последовательности проведения документов есть существенный недостаток: она может длиться от нескольких минут до нескольких часов, а это во многих случаях не совсем приемлемо.

Однако есть путь, который позволяет не восстанавливать.последовательность документов. Для этого нужно разрешать правку документов только текущим числом, а переделывать документы, которые были выписаны раньше (до сегодня), *сторными операциями*. Сторно — это операция, которая аннулирует действие другой операции. Например, была сделана проводка выдачи в подотчет из кассы 100 р.: 71 / 50 / +100-00. Тогда сторная операция для нее будет выглядеть так: 71 / 50 / —100-00.

Допустим, следует переделать расходную накладную. Для этого следует выполнить следующие действия:

- оформить возврат товара, полученного по неверной накладной,
- выписать новую накладную (копию старой с изменениями).

Таким образом, проблема «заднего числа» исчезает. Однако и этот метод переделки документов имеет недостаток: иногда невозможно разобраться, кто и зачем делал правки; многие клиенты бывают недовольны появлением лишних возвратных документов в актах сверки. Эти проблемы можно решить через создание документа Переделка, который автоматизирует процесс сторных операций и выписки правленых документов. Строить программные защиты от отрицательных остатков не только неэффективно, но и невозможно. Дело в том, что правка задним числом может и не вызвать появление отрицательных остатков, а ошибка будет. Чем больше остатки в базе не сходятся с реальными, тем больше недоверия к программисту, причем необоснованного.

К сожалению, в реальности запретить работать задним числом практически невозможно. В настоящий момент не выработано приемлемой схемы работы в предыдущем периоде. Можно рекомендовать лишь один способ борьбы с перечисленными выше ошибками: если возникло несоответствие компьютерного остатка реальному, следует проводить инвентаризацию.

Работа с константой ДатаЗапретаРедактирования

Константа ДатаЗапретаРедактирования определяет дату, до которой нельзя производить изменения документов. Одной из проблем, возникающих при работе с этой константой, является то, что пользователь забывает возвращать ее на место (изменил константу на ту дату, где он хотел поработать, и забыл запре-

тить доступ к этому периоду в дальнейшем). Для того чтобы пользователь не забывал возвращать константу на нужную дату, можно изменить процедуру ПриНачалеРаботыСистемы так, чтобы она сама устанавливала нужную дату запрета редактирования:

```
Процедура ПриНачалеРаботыСистемы()
Если не(константа.ДатаЗапретаРедактирования = РабочаяДата() - 1)
  Тогда константа.ДатаЗапретаРедактирования =» РабочаяДата() - 1;
КонецЕсли;
КонецПроцедуры
```

В некоторых случаях нужно запретить не все документы, а только товарные (влияющие на остатки товара). Тогда нужно ввести еще одну константу: ДатаЗапретаОстатков. Процедура ПриНачалеРаботыСистемы будет выглядеть так:

```
Процедура ПриНачалеРаботыСистемы()
Если не(константа.ДатаЗапретаРедактирования = РабочаяДата() - 2)
  Тогда константа.ДатаЗапретаРедактирования = РабочаяДата() - 2;
КонецЕсли;
Если не(константа.ДатаЗапретаОстатков = РабочаяДата () - 1) Тогда
    константа.ДатаЗапретаОстатков = РабочаяДата() - 1;
КонецЕсли;
КонецПроцедуры
```

Но это еще не все. Нужно также запретить запись, проведение, удаление и снятие с проведения документов. В типовой «Торговле» 8.0 во всех документах при проведении вызывается процедура МожноЗаписатьДокумент. В новых конфигурациях эта процедура называется глМожноЗаписатьДокумент. В коде это выглядит так:

```
Функция МожноЗаписатьДокумент(Конт) Экспорт
Если Конт.ДатаДок <* Константа.ДатаЗапретаРедактирования Тогда
    Предупреждение("Нельзя записывать документы с датой,
        | более ранней, чем Константа
ДатаЗапретаРедактирования!");
    Возврат 0;
КонецЕсли;
Если (Конт.Вид() = "РасходнаяНакладная") или                    \
     (Конт.Вид() = "Перемещение") или
     (Конт.Вид() щ "ПриходнаяНакладная") или
     (Конт.Вид() = "Списание") Тогда
    Если Конт.ДатаДок <= Константа.ДатаЗапретаОстатков Тогда
        Предупреждение("Нельзя записывать документы с датой,
            I более ранней, чем Константа ДатаЗапретаОстатков!");
        Возврат 0;
    КонецЕсли;
КонецЕсли;
Возврат 1;
КонецФункции
```

С проведением и записью разобрались. Осталось удаление и снятие с проведения. Для удаления и отмены проведения существуют предопределенные процедуры в глобальном модуле ПриОтменеПроведенияДокумента и ПриУдаленииДокумента. Эти функции выглядят так:

```
Процедура ПриОтменеПроведенияДокумента(Докум)
Если Докум.ДатаДок <= Константа.ДатаЗапретаРедактирования Тогда
```

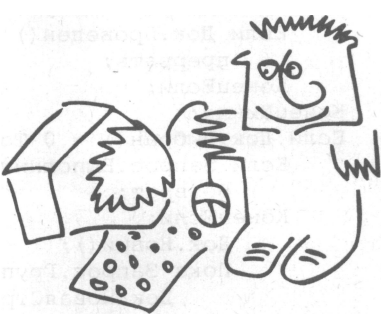
```
        Предупреждение("Нельзя снимать документы с проведения,
          выписанные раньше Константы ДатаЗапретаРедактирования!");
        СтатусВозврата (0);
        Возврат;
    КонецЕсли;
    Если (Конт.ВидО = "РасходнаяНакладная") или
         (Конт.ВидО = "Перемещение") или
         (Конт.ВидО = "ПриходнаяНакладная") или
         (Конт.ВидО = "Списание") Тогда
        Если Конт.ДатаДок <= Константа.ДатаЗапретаОстатков Тогда
            Предупреждение("Нельзя записывать документы с датой,
              более ранней, чем Константа ДатаЗапретаОстатков!");
            Возврат 0;
        КонецЕсли;
    КонецЕсли;
    КонецПроцедуры

    Процедура ПриУдаленииДокумента(Докум,Режим)
    Если Докум.ДатаДок <= Константа.ДатаЗапретаРедактирования Тогда
        Предупреждение("Нельзя удалять документы, выписанные раньше
          Константы ДатаЗапретаРедактирования!");
        СтатусВозврата(0);
        Возврат;
    КонецЕсли;
    Если (Конт.ВидО = "РасходнаяНакладная") или
         (Конт.ВидО = "Перемещение") или
         (Конт.ВидО = "ПриходнаяНакладная") или
         (Конт.ВидО = "Списание") Тогда
        Если Конт.ДатаДок <= Константа.ДатаЗапретаОстатков Тогда
            Предупреждение("Нельзя записывать документы с датой,
              более ранней, чем Константа ДатаЗапретаОстатков!");
            Возврат 0;
        КонецЕсли;
    КонецЕсли;
    КонецПроцедуры
```

И наконец, нужно не забыть запретить редактировать константу некоторым пользователям, иначе запрет потеряет смысл.

Журнал регистрации

Журнал регистрации отслеживает события, происходящие с базой данных. Например, в журнал заносится факт изменения конфигурации, документов или справочников. Правда, в журнале не фиксируется, какие конкретно изменения были внесены.



По журналу можно определить, какой пользователь изменял (проводил, записывал, открывал) документ, дату и время изменения, но не более того. Если возникает желание определить, кто изменил конкретную позицию в

документе, то решить эту проблему можно только анализируя резервные копии базы данных.

Следует отметить, что при использовании распределенной базы данных в каждом филиале находится свой журнал регистрации, и журналы разных точек не консолидируются в общий при обмене данных.

В журнал регистрации по умолчанию заносятся только действия пользователей. Действия обработок или отчетов, влияющих на структуру справочников или документов, в журнал не заносятся. Если программист желает, чтобы обработки, изменяющие документы, оставляли след в журнале регистрации, то в программах следует использовать функцию ЗаписьЖурналаРегистрации.

Резервирование товара

Кроме расходных накладных, которые влияют на остатки товара, операторы могут выписывать счета на оплату, которые влияют не на остатки товара, а на резервы товара.

Обычно счет выписывается на несколько дней, после чего он должен обнулиться, а товар должен сняться с резерва. Проблему ликвидации просроченных счетов можно решить автоматической обработкой, создавая каждый день при первом входе в систему документ СнятиеРезерва, например, следующей процедурой:

```
Процедура ПриНачалеРаботыСистемы()
Перем Запрос, ТекстЗапроса;
// Держать резерв 3 дня.
ВыбДата = ПолучитьДатуТА();
Дни = ВыбДата - 3;
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(Сформировать)
|Период с ВыбДата по ВыбДата;
|Счет = Регистр.РезервыТоваров.ПоСчету;
|Дат = Регистр.РезервыТоваров.ПоСчету.ДатаДок;
|Ост = Регистр.РезервыТоваров.РезервТовара;
|Функция КонОст = КонОст(Ост);
(Группировка Счет;
|Условие(Дат < Дни);
|"//}}ЗАПРОС;
Док = СоздатьОбъект("Документ.СнятиеРезерва");
Док.ВыбратьДокументы(РабочаяДата(),РабочаяДата());
Пока Док.ПолучитьДокумент() = 1 Цикл
    Если Док.Проведен() = 1 Тогда
        прервать;
    КонецЕсли;
КонецЦикла;
Если Док.Выбран() = 0 Тогда
    Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
        Возврат;
    КонецЕсли;
        Док.Новый();
        Пока Запрос.Группировка(1) = 1 Цикл
            Док.НоваяСтрока();
            Док.ПоСчету = Запрос.Счет.ТекущийДокумент();
```

```
    КонецЦикла;
        Док.Записать();
        Док.Провести();
        сообщить("Создан документ снятие резерва №" +
Док.НомерДок + " от " + Док.ДатаДок);
КонецЕсли;
КонецПроцедуры
```

Контроль отрицательных остатков

На рис. 5.1 показаны несколько стадий обработки документа. Цифрами обозначены операции следующих работников:

1 — заказчик,

2 — оператор, выписывающий товарные документы,

3 — работники склада,

4 — работники службы доставки (экспедиторы и т. д.).

Рис. 5.1. Схема движения товара

В приведенной схеме задействовано четыре человека. Там, где работают люди, возникают проблемы понимания. Даже два человека не всегда правильно понимают друг друга, что же можно сказать о цепочке, составленной из большего количества людей. Выписанный документ на любом этапе обработки может быть скорректирован, что потребует взаимодействия работников всей цепочки, а значит, появятся ошибки.

Изменение документов может привести к ошибкам

Изменение уже проведенных товарных документов (как приходных, так и расходных) может привести к возникновению отрицательных остатков. Рассмотрим пример (рис. 5.2).

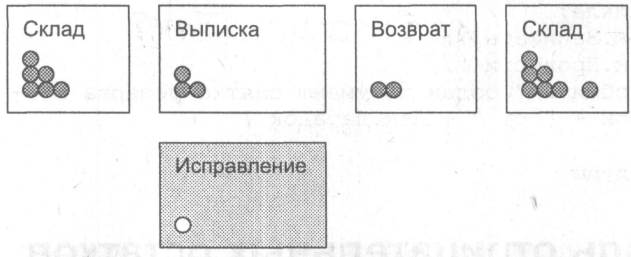


Рис. 5.2. Изменение документов задним числом может привести к ошибкам

1. На складе было 6 единиц товара.
2. Выписали 3 единицы.
3. Вернули 2 единицы.
4. Осталось 5 единиц.
5. После этого произвели правку выписанного количества, заменив 3 на 1. В результате правки остаток на складе будет равным 7.

По какой причине была произведена правка, была ли она санкционирована или нет, не известно.

Этот пример показывает, как могут появляться товары в базе данных, которых на самом деле нет. Оператор, выписывающий товар, ориентируется на информацию базы данных, а не на реальные остатки склада.

Возможны несколько вариантов возникновения отрицательных остатков. Например, оператор выписывает 7 единиц товара, в то время как на складе их только 5. Документ правится с 7 на 5, и в базе остается 2 единицы товара, которые могут в следующий раз опять выписать. Возникновение отрицательных остатков возможно и при удалении документа возврата.

Какие есть варианты решения проблемы отрицательных остатков?

- Запретить проводить документы задним числом, что приведет к невозможности отдать покупателю документ с тем же номером, но другим количеством.
- Переделывать документы путем сторнирования неправильно выписанных. На расходную накладную выписывается сторная накладная (расходная накладная с минусами), а потом вновь выписывается правильный документ. В этом случае возникают лишние документы, которые в одних отчетах следует учитывать, а в других — нет.
- Периодически восстанавливать последовательность документов. Таким образом можно будет отловить все возникшие в ходе работы минусы. Однако если ошибка была совершена, а минусов не возникло, то данным способом проблему не решить.
- Проведение ревизий на складе. В этом случае проблема выходит из сферы компетенции программиста. Конечно, проведение инвентаризации тормозит работу предприятия, зато в ходе инвентаризации можно определить сумму недостачи и высчитать ее из материально ответственных лиц.

Таким образом, лучшим решением проблемы возникновения нереальных остатков будет проведение ревизии.

Из чего состоит база данных?

База данных состоит из следующих файлов:

- 1Cv7.DD — файл структуры базы данных,
- 1Cv7.MD — файл конфигурации,
- *.dbf — файл базы данных,
- *.cdx — индексный файл,
- *.dll — дополнительные функции.

Если уничтожить файлы *.cdx, то при следующем запуске программы они будут восстановлены без потери каких-либо данных.

Если уничтожить файлы *.dll, то некоторые функции, например работа с Интернетом, перестанут работать.

Если уничтожить файлы *.dbf, то при следующем запуске системы программа восстановит их, однако все данные окажутся потерянными.

Если удалить файлы 1Cv7.DD или 1Cv7.MD, то программа перестанет работать.

Если удалить каталог userdef, то программа забудет про пользователей и их пароли.

Если удалить каталог syslog, то программа очистит журнал работы с программой (кто когда заходил, какие документы создавал или уничтожал).

Если удалить каталог extforms, то программа забудет про внешние подключаемые формы и меню Сервис • Дополнительные возможности окажется пусто.

Релизы и редакции 1С-программ

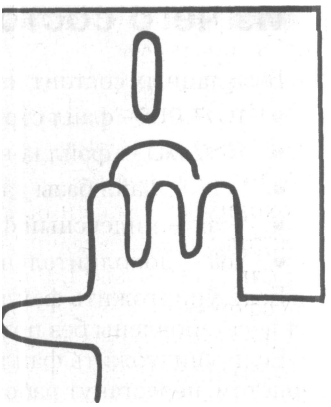
Срок жизни *типовой редакции* (то есть той конфигурации, которая поддерживается фирмой «1С») составляет 1-2 года. За это время фирма «1С» выпускает несколько десятков релизов.

Релиз — небольшая модификация конфигурации, связанная с исправлением имеющихся ошибок, выходом новых форм документов и отчетов, небольшими изменениями в законодательстве.

Редакция подразумевает коренное изменение структуры и методологии программы. То есть необходимость в новой редакции возникает, когда выявленные ошибки или веяние времени делают невозможной реализацию новых идей старыми средствами. Появление новых редакций связано с существенными изменениями в законодательстве (изменение Плана счетов, введение налогового учета) и потребностью коренных изменений в структуре данных и выполняемых функций.

Механизмы обновления программ

При переходе от релиза к релизу применяются три способа обновления конфигурации: «Загрузка измененной конфигурации», «Объединение конфигураций» и перетаскивание объектов из одной конфигурации в другую операциями Drag-and-Drop.



- *Загрузка измененной конфигурации* применяется, если последующая конфигурация является потойком изменяемой конфигурации. То есть конфигурация была скопирована, изменена и загружена обратно. Если обе копии конфигурации подверглись изменениям, то при попытке загрузить одну из них в другую система выдаст предупреждение: «Выбранный файл конфигурации не является потомком данного файла. При реструктуризации может произойти разрушение данных».
- *Объединение конфигураций* применяется, если требуется объединить две разные конфигурации (когда-то бывшие одной). При объединении конфигураций можно отдать приоритет либо текущей, либо загружаемой конфигурации и выбрать режим замещения или объединения объектов.
- Для *перетаскивания* объектов из одной конфигурации в другую следует открыть рядом Конфигураторы двух баз и мышью перетащить нужные объекты из одного Конфигуратора в другой. Также можно использовать для этих целей буфер обмена: объект копируется в одном месте, а затем вставляется в другом.

При переходе от редакции к редакции проектируются специальные конверторы, которые осуществляют перенос данных между двумя информационными базами, например в формате XML.

Как перенести базу данных за пределы офиса заказчика?

В зависимости от сложности задач программист может взять «на вынос» или всю базу данных, или только самые необходимые для работы файлы.

Если вы хотите взять базу данных в минимальной конфигурации, например для работы в офисе, то следует скопировать файлы 1Cv7.DD или 1Cv7.MD. Для запуска базы данных следует запустить Конфигуратор, указав путь к каталогу, в котором находятся эти файлы.

Если требуется взять всю базу данных, то следует заархивировать весь каталог базы данных или воспользоваться Конфигуратором (Администрирование • Сохранение данных).

Кстати, вынося базу данных за пределы офиса заказчика, помните, что вы берете на себя ответственность за попадание базы данных в третьи руки.

Как вносить изменения в типовую конфигурацию?

Первое правило — не торопитесь писать программы по первой просьбе клиента. Многие проблемы у пользователя возникают из-за незнания всех возможно-

стей типовой конфигурации. Возможно, изменения, которые просит сделать заказчик, противоречат законодательству или решаются иными, чем он предложил, способами. Выясните, что пользователь сэкономит в результате внедрения программы. Заставьте клиента убедить вас в необходимости внесения изменений. Конечно, этот совет может противоречить принципам, по которым работает программист: раз позвали, значит, это нужно. Однако, если вы будете делать только необходимые изменения, вас будут знать и рекомендовать не как ремесленника, а как толкового специалиста.

Если же все-таки пользователь настаивает на внесении изменений, то попросите сформулировать изменения в письменном виде. Задание следует как можно больше детализировать: если вам написали проводки документа, то следует уточнить аналитику проводок; если попросили добавить документ, следует уточнить реквизиты документа, возможные движения регистров, проводок и печатную форму, конечную цель документа. Кстати, создание нового документа не всегда оправдано. Перед созданием в Конфигураторе нового документа подумайте, сможете ли вы решить задачу обработкой или отчетом. Если документ не содержит печатной формы, не является первичным документом (таким как приходный кассовый ордер или накладная) и не должен использоваться в отчетах (например, в реестре документов), то использование внешней обработки более предпочтительно по сравнению с созданием нового документа.

Для начала изменения следует производить в копии базы данных. Все изменения надо тщательно документировать. Изменяемый код не удаляйте, а помечайте как комментарий. В комментариях также следует ставить дату изменений, фамилию программиста и причину внесения изменений. Изменения также можно вносить в описание конфигурации или в описание документов или внешних отчетов.

Перед изменением программы реальной базы данных работоспособность внесенных изменений должна проверяться пользователем в копии. Только конечный пользователь может определить правильность работы программы.

После проверки работоспособности программы ее можно переносить в реальную базу данных. Для этого следует зайти в Конфигуратор базы данных и сделать резервное копирование с указанием даты и времени сохранения, и лишь после этого выполнить загрузку измененной конфигурации.

Может быть, описанная выше схема может показаться долгой и нудной, но поверьте, что она выверена, как устав караульной службы в армии. Внести изменения легко, а исправить возможные ошибки бывает почти невозможно. В случае, если программа не была проверена или нет возможности откатить систему на момент «до вмешательства программиста», то винить, кроме как программиста, больше некого.

Сохранение базы данных

Сохранение базы данных делается механизмами «1С» (команда Конфигуратор • Администрирование • Сохранить данные). По умолчанию сохраняются только файлы самой конфигурации, данные и список пользователей.

Журнал регистрации и каталог ExtForms при сохранении базы данных не сохраняются. Для того чтобы файлы из каталога ExtForms сохранялись, следует добавить в файл 1cv7file.lst, который находится в каталоге .AProgram Files\1Cv77\BIN, строку Uextforms* . *.

Для сохранения журнала регистрации определен специальный механизм в самом журнале (команда Монитор • Монитор • Архивирование журнала).

Проблемы, возникающие при открытии базы данных

Прежде всего нужно определить, какие сообщения выдает программа. Может быть, программа требует загрузиться в монопольном режиме или сообщает, что каталог пользователя занят, то есть кто-то уже зашел в программу под выбранным именем.

Если программа ничего не говорит, тогда следует проверить, запускается ли Конфигуратор. Конфигуратор может не запускаться по нескольким причинам.

1. База находится на сетевом ресурсе, доступном только для чтения. Файлы «1С» должны быть доступны для записи, так как для начала программа должна сделать запись в журнал регистрации и обновить индексные файлы.
2. База находится на сетевом ресурсе, а сетевой ресурс позволяет открыть по сети ограниченное количество файлов. Проверить это можно, закрыв программу у какого-то пользователя и открыв у проблемного. Данная проблема проявляется на компьютерах с Windows 98.
3. Поврежден файл журнала регистрации 1cv7.mlg (файл находится в каталоге syslog). Нужно его удалить или разобраться в структуре записи и устранить неисправность вручную.
4. Если база копировалась с компакт-диска, возможно, вы забыли снять с файлов атрибут «только чтение».
5. Если вы склонны считать, что проблема в работоспособности сети, скопируйте базу данных на локальный компьютер.
6. На диске должно быть не менее 200 Мбайт свободной памяти. Для комфортной работы лучше, чтобы у вас было хотя бы 500 Мбайт.

Иногда в случае непонятных сбоев в работе программы решением может быть копирование базы данных в другой каталог или перезагрузка компьютера.

Если и такие методы не устраняют проблемы, проверьте загруженные компоненты. Если вы работаете в «Торговле», то должен быть загружен оперативный учет, если в «Бухгалтерии» — бухгалтерский и т. д. Если у вас возникли проблемы с загрузкой компонентов, то либо ваша программа взломана (причем неправильно), либо что-то не так с ключом (не доступен по сети, нерабочий разъем и т. д.) и требуется переустановка программы или устранение проблем, связанных с ключом.

Лечение базы данных

В механизмах «1С» предусмотрена специальная процедура тестирования и исправления ошибок, возникающих, например, когда в момент проведения документа компьютер с базой данных зависает. Часть регистров в этом случае может измениться, а часть — нет. Чтобы исправить ошибки, следует запустить в Конфигураторе в меню Администрирование команду Исправление и тестирование БД.

Однако возможны случаи, когда тестирование не помогает. Тогда следует сделать копию файлов 1 Cv7.DD и 1 Cv7.MD в другой каталог и загрузить из этого каталога Конфигуратор, а затем и базу. Если запустить программу из каталога, где находятся файлы 1Cv7.DD и 1Cv7.MD, но нет DBF-файлов, программа создаст структуру пустых DBF-файлов.

Далее следует протестировать базу (Конфигуратор • Администрирование • Тестирование и исправление ИБ). Если программа после этого запускается (то есть файлы конфигурации 1Cv7.MD и 1Cv7.DD работают без сбоев), то следует вручную анализировать базу данных. Для того чтобы определить, какой из DBF-файлов является источником сбоя, следует порциями копировать файлы из сбойной базы во вновь созданную базу. Каждый раз после копирования следует перезапускать базу данных. Допустим, что в результате проведенных действий был найден сбойный файл, например dh1160.dbf. Попробуйте скопировать этот файл из сохраненной ранее базы данных.

Если база данных не запускается на этапе переноса файлов 1 Cv7.DD и 1 Cv7.MD, то это значит, что виноват один из этих файлов. Для выявления источника сбоя * следует последовательно удалять из конфигурации объекты: сначала документы, затем справочники, журналы и т. д.

Поиск причин возникновения ошибок

Ошибки могут быть разными: мнимыми и реальными, связанными с незнанием системы, преднамеренными, случайными, систематическими и разовыми; ошибками пользователя, ошибками программиста, ошибками операционной системы или разработчиков.

Когда бухгалтер заявляет вам: «Я знаю, что этот клиент ничего не должен, а ведомость по взаиморасчетам показывает наличие долга. Пожалуйста, исправьте программу», — такое утверждение нельзя просто принимать на веру. Оно свидетельствует не более чем о некомпетентности бухгалтера. Цифры не могут попасть в базу данных сами по себе. Правильность результатов можно проверить карточкой счета или другим отчетом, который выдает ссылки на первичные документы.

Я бы советовал подходить ко всем стандартным отчетам (карточка счета, анализ счета, оборотно-сальдовая ведомость PI др.) с той точки зрения, что отчеты работают верно. Этот совет будет полезен программисту-новичку, который прежде всего ищет ошибки в своей программе, а не там, где они обычно бывают.

Один из вариантов поиска ошибок состоит в сверке компьютерных отчетов с их бумажными версиями, сделанными некоторое время назад. Кстати, вместо бумажных копий можно сохранять отчеты в XLS- или MXL-формате.

Существует класс ошибок, когда остатки на конец периода не совпадают с остатками на начало периода. Например, сальдо конечное по счету «Касса» за 01.12.03 не идет с сальдо начальным по счету «Касса» за 02.12.03. В этом случае требуется протестировать базу данных. Для этого следует выполнить следующие действия:

1. Сохраните базу данных.
2. Протестируйте ее, выполнив команду Конфигуратор • Администрирование • Тестирование и исправление БД.

Если тестирование не изменило ситуацию, то можно попробовать изменить версию программы, например заменить версию 20 на 21 (не следует путать версию 1С-программы и версию конфигурации). Для того чтобы определить версию 1С-программы, следует в Конфигураторе выбрать пункт меню Помощь • О программе. Версия 1С-программы будет указана в скобках в правом верхнем углу, например: 7-70-0020.

Поиск ошибок может производиться сравнением копий баз данных, сохраненных в разные дни. Для этого следует наладить регулярное архивирование базы данных. Лучше всего базы данных за различные дни архивировать на компакт-диски.

Проблемы в распределенных базах данных

Компонента «Распределенная Информационная база» позволяет синхронизировать базы данных, не объединенные одной локальной сетью. Изменения, сделанные на одной из баз через систему обмена данными с центральной базой, через некоторое время попадут в другие распределенные базы данных.

Важно помнить, что при использовании распределенной базы приоритетом обладает центральная база. Поэтому, если документ был одновременно изменен на периферийной базе данных и, допустим, перепроверен в центральной, то изменения, сделанные на периферийной базе данных, будут вытеснены изменениями, сделанными в центральной базе.

Чтобы избежать неприятных сюрпризов, следует запретить всем, кроме центра, изменять некоторые справочники, например справочник Номенклатура.

Однако такой способ решения проблемы не годится для справочника Контрагенты. Проблему двойных клиентов можно решить установкой реквизиту ИНН свойств Отбор по реквизиту и Сортировка. В этом случае поиск клиента можно производить не по наименованию (которое может иметь несколько вариантов: «ООО Прогресс», «Прогресс», «Прогресс ООО»), а по уникальному для каждого предприятия коду — ИНН.

При работе с распределенной базой следует внимательно следить за префиксами справочников и документов. Если после нескольких обновлений появились карточки товаров или расходные накладные с одинаковыми номерами, это значит, что алгоритм назначения кодов справочников не работает. Для назначения номера используется команда УстановитьНовыйНомер (Префикс).

Переход на новые версии программы

Не спешите менять версии программы торгового учета (речь идет только о программах торгового учета, а не о «Бухгалтерии»). Новая версия лучше, удобнее, но обычно требует больших вычислительных ресурсов. Я бы советовал выждать полгода или год после того, как появится новая версия программы, и лишь после этого переходить на нее.

Глава 6
Психология общения с клиентом

В этой главе описываются психологические приемы общения программистов с клиентами. Разделы содержат практические наблюдения и рекомендации.

Могу сказать, что если бы я сколько-то лет назад усвоил описанные здесь особенности общения с людьми, то я сэкономил бы вагоны времени и сотни километров нервов.

Психологический прием «Тук-тук — перестук»

С детства известно правило: ябедничать нельзя. Как это ни парадоксально, но на систему «программист — заказчик» это правило не распространяется. Клиент — многоголовый дракон. Одна голова заключает договор, от другой зависит оплата вашего труда, а с остальными приходится общаться в процессе работы. Как правило, именно в последнем случае частенько возникают сложности. Но программиста должно интересовать, как сделать работу максимально быстро и с наименьшими затратами, а не как решить психологические проблемы, возникшие в ходе общения с подчиненными заказчика.

Я опишу психологический прием, который имеет успех в 90 % случав. Назову его «Тук-тук — перестук».

Выберите ответ на вопрос: «Что вы будете делать, если кто-то (из одноранговых товарищей) не хочет выполнять ваши решения?»

1. Я сам не могу принимать решения. Решения принимает начальник. Я могу лишь подготовить решения или высказать мнение.
2. Следует убедить человека.
3. Следует настучать на человека начальнику и попросить начальника воздействовать на своего подчиненного.

Если вы выбрали один их первых двух вариантов, то вы обрекаете себя на вечные согласования и хождения вокруг да около проблемы. Я настаиваю на третьем варианте! Когда-то мне повезло и я съездил в США, где посмотрел, как делается бизнес американцами. Меня поразила там одна вещь. Если работник не знает, как поступить с клиентом, он просит клиента обратиться к менеджеру. Не решайте проблему сами, а отправляйте к тому, кто наверху! Использовать начальника как козырную карту в споре — очень сильный ход.

Подумайте, можете ли вы сказать о себе: «Хочешь — научим, не можешь — поможем, не хочешь — заставим». Если ваше нутро противится такому принципу, то не оказывается ли так, что вы играете роль козла отпущения?

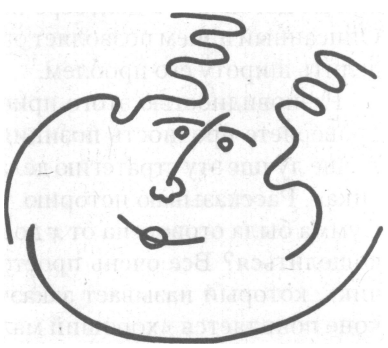
Снова примеры из практики.

Работник беспокоил по пустякам. После обращения к директору с фразой: «Как вы думаете, правильно ли я считаю, что если постоянно возникают конфликты по поводу недоделок программы, то или программист не умеет работать, или принявший работу специалист не тот, за кого себя выдает? Прошу вас помочь разобраться» и пятиминутного разговора вызовы прекратились.

Одна дама долго надоедала программисту тем, что не могла прочитать сообщение программы о том, что приложение следует запустить в монопольном режиме. После привлечения к данному делу начальника дама сказала, что «раньше программа выдавала другую табличку». Однако после разговора проблемы как-то исчезли, а вскоре дама уволилась.

Возникла заминка с разносом документов, и работник «включал дурака», когда нужно было разносить определенный тип документов. После обращения к начальнику с просьбой разъяснить подчиненному, как выполнять «вот эту простую операцию», дурак перестал включаться.

Психологический прием «Прикинься дурачком»



Когда подчиненный «тупит», то следует стремглав обращаться к его начальнику, а если начальник «тупит», то его нужно убедить в том, что эта работа не для него и что вы с радостью объясните подчиненным, как справляться с затруднениями.

Кстати, иногда следует самому прикинуться дурачком. Кому из нас не нравился Коломбо — придурковатый следователь, которого никто не принимает всерьез до заключительного аккорда? Коломбо самым простецким образом выводил на чистую воду толпы злоумышленников. Вот и я предлагаю вам попробовать вывести собеседников на чистую воду по методу Коломбо.

Приведу примеры. Случай первый. Вас просят автоматизировать бухгалтерский учет сети магазинов. Согласно подходу «Прикинься дурачком» следует попытаться убедить клиента в том, что автоматизация ему не нужна, что это будет дорого стоить, будет долго внедряться, придется уволить часть работников и заключить договор на обслуживание. Если вы не сможете убедить собеседника, а, наоборот, собеседник убедит вас — это здорово. Если убедили вы, то еще лучше: вы избавитесь от ненадежного клиента. Во время процесса убеждения следует внимательно вслушиваться в аргументы партнера. После беседы следует записать систему убеждений клиента и с толком использовать ее в дальнейшей работе. Полезно иметь в кармане диктофон, а в рукаве — микрофон к нему. Потом здорово все послушать еще разок и сделать выводы заново.

Случай второй. Нужно ввести в структуру базы данных новый тип документов. Заказчик настаивает на реконфигурации, а вы говорите, что он вас не убедил и что можно попробовать обойтись без нового документа, что уже имеющихся документов и так достаточно для работы. Вы предлагаете вновь и вновь пройтись по всей цепочке аргументов и попробовать найти в ней дыры. Мол, «чувствую, что что-то тут не так, а что — не могу определить».

Можно и просто сказать: «Извините, я сегодня плохо соображаю, ничего не могу понять, и вообще мне кажется, что вы сказали бессвязный набор слов». Хотя эти фразы не для слабонервных, а для людей сильных духом и уверенных в своем интеллекте. Ну кто из нормальных людей может просто взять и признаться в том, что у него плохо с головой? Если клиент и после этих «сильных духом» фраз продолжает держать вас за грудки и настаивает на введении нового документа, то можно мирно так — именно как делает Коломбо! — сказать: «Знаете, что-то это все меня не убеждает. Не найдется ли у вас еще каких-нибудь дополнительных аргументов?» Ключевым является слово «дополнительный». Оно во многих случаях работает превосходно[1].

Читатель может подумать, что автор несет бред: меняться местами с заказчиком — придумают же такое! Но залог долгого и плодотворного сотрудничества не в том, что заключаются договоры, а в том, что есть необходимость совместной работы. Найти работу, за которую не платят, — просто. Программисту нужны не любые договоры, а долгосрочные; не любые клиенты, а те, кто исправно платит. Описанный прием позволяет определить заинтересованность клиента в вас и определить широту его проблем.

Разновидностью этого приема может быть прием «Лезь на рожон», когда вы проверяете прочность позиции противной стороны методом прямого давления. А еще лучше эту стратегию делать при помощи приема «хорошего и плохого мальчика». Рассказываю историю. Был заключен договор на производство веб-сайта. Сумма была оговорена от *x* до *x*5*. Как проверить, насколько заказчик готов раскошелиться? Все очень просто. Из вашей команды выскакивает «плохой мальчик», который называет заказчику сумму *x*5*. Заказчик в шоке, и тут на белом коне появляется «хороший мальчик», который все решает и устанавливает справедливую цену. (Во времена сталинских репрессий следователи применяли подобную стратегию работы с подсудимыми. Один из следователей был жестоким, а другой — мягким. Первый наводил жуть, а второй пожинал плоды — записывал признания.)

Вообще, групповой способ убеждения очень эффективен. Даже если во время встречи участвуют двое против одного и один из тех двоих просто надувает щеки, по примеру Кисы Воробьянинова, — даже в этом слабом варианте преимущество будет на стороне тех, кого двое.

Вы прочитали все это и решили, что это не для вас? Ведь ваша работа связана с серьезностью? Вы не можете выглядеть несерьезно в глазах окружающих? Тогда статья именно для вас!

Однако имейте в виду, что, прикидываясь дурачком, предприниматель ходит по лезвию ножа. Так можно и правда убедить клиента, что ты дурак.

Психологический прием «Позволь себе не называться программистом»

Программисту наверняка знакома ситуация, когда он сиднем просиживает над одной проблемой на протяжении дня и не может ее решить. Вопроса, а не обратиться ли за «помощью зала», не возникает, потому что если человек обращается за помощью, то он расписывается в своем бессилии.

Как быть? Ответ прост: разрешить называть себя не *программистом,* а, скажем, *специалистом по посещению клиентов*. Непонятно? Если вы не программист, то разве можете выполнять программы сложнее ста строк (я где-то читал, что программист должен писать в день около ста строк)? Если вам не удается в короткие сроки справиться с задачей, то вам следует обращаться за помощью. Выскажу крамольную мысль: любая программа пишется за полдня. (Конечно, есть и исключения: некоторые программы пишутся за пять минут.)

Такой подход стимулирует внутригрупповое взаимодействие и дает выигрыш в разделении труда. Один программирует, другой посещает клиентов и следит за психологической обстановкой, третий ищет новых заказчиков.

Ярлыки помогают, но и тормозят движение вперед. Попробуйте, так, ради смеха, примерить шляпу агента, финансового аналитика, бухгалтера, руководителя.

Психологический прием «Используй структуру клиента»

Одна из распространенных ошибок начинающего программиста заключается в неверном понимании технологии работы с клиентами. Исполнители и администраторы, отсчитывающие деньги, — это, скорее всего, разные люди. Администратор может быть не в курсе всех дел, и он делегирует полномочия своему подчиненному, который проверяет работу. Конкретный же работник не принимает решений оплачивать или не оплачивать счета на обслуживание.

Пример использования структуры клиента. Сделана программа. Программист подходит к директору и просит заплатить за работу. Директор говорит, что он не занимается проверкой работоспособности и отсылает от себя. Программист идет к исполнителям и просит заплатить. Те с недоумением смотрят на него и говорят, что они не могут, это не их проблема. Создается впечатление, что организация не хочет оплачивать выполненную работу. Это впечатление ложное. Для скорейшего разрешения такой ситуации следует разбить задачу на две:

1. Добиться подтверждения факта выполненных работ. Подтверждение получается у исполнителей. Для этого подписывается акт выполненных работ, ставятся росписи в журнале потраченного времени.
2. Получить подтверждение желания оплаты. Для этого следует подходить к руководителю с подписанными актами.

Вещь, описанная выше, проста, но порой ставит новичков в тупик.

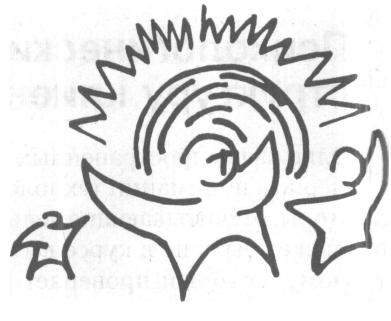
Еще один случай из практики использования структуры клиента. Определенная программа делается в течение длительного времени. Клиент постоянно

вносит изменения в алгоритм. Вследствие этого завершение работы над программой затягивается и у руководства создается впечатление, что программист — не специалист, за которого он себя выдает, а просто какой-то шарлатан. Постепенно растет напряженность и нервозность. Как быть?

Одним из возможных решений может быть организация круглого стола, то есть встречи, где каждый выразил бы свои пожелания по поводу выхода из создавшегося положения. По ходу встречи должен вестись протокол с указанием пожеланий заказчика. После встречи формируется техническое задание и определяются сроки внедрения. После окончания сроков вновь назначается встреча, на которой проводится «разбор полетов» и определяются действительные виновники простоя.

Важная часть в этой стратегии заключается в том, что руководитель самостоятельно должен принять решение. Например, на одной встрече было решено обновить сервер. Наступает время новой встречи, а фонды на «перевооружение» не выделены. Значит, виноват не программист, а тот, кто тянет с решением выделить средства.

Психологический прием «Я вас правильно понял?»



Когда военный получает приказ, он должен выполнить следующие вещи:

- повторить приказ,
- уяснить приказ,
- выполнить рекогносцировку,
- спланировать время.

Когда я впервые услыхал от офицера на военной кафедре этот список действий, я был поражен тупоумием армейцев: это надо же, они должны «уяснить приказ» (!), как будто это не очевидно. Однако если вспомнить, что у военных за каждой строкой устава лежат горы трупов, то впечатление меняется...

А теперь реальные примеры неправильно уясненных приказов. Человеку было велено прийти на встречу в начале двенадцатого. Его ждали с одиннадцати часов (потому что начало двенадцатого понималось как одиннадцать часов с копейками), а он пришел в двенадцать, потому что услышал и запомнил слово «двенадцать».

Пример номер два (реальный случай из моей практики). Руководитель группы монтажников говорит: «Проложи кабель от двери до противоположной стены». Подчиненный берет под козырек — и прокладывает кабель поперек комна-;гы, а не по плинтусу, как хотел (но не сказал!) руководитель.

Пример номер три. Ведущий программист поручил подчиненному написать программу. Тот подумал, что должен выполнить всю работу сам, но застопорился на каком-то этапе. Первый, вместо того чтобы проконтролировать выполнение, ждал, пока подчиненный сам скажет о своих проблемах. В результате программа так и не была написана.

Вы можете дополнить этот список? Я думаю, причина подобных нестыковок в различных моделях мира договаривающихся сторон. Слова говорят одинаковые, а понимают их по-разному.

Попробуйте сблизить модели, задав уточняющий вопрос:

- Я правильно понял, что должен быть у тебя в двенадцать?
- Я правильно понял, что программу мне следует делать самому по принципу «ни шагу назад»?
- Я правильно понял, что нужно тянуть кабель напрямую, чтобы сэкономить пару метров?

Используйте фразу «Я правильно тебя понял?» сами и просите собеседника объяснить, как он понял вас. Особенно хорошо будет, если собеседник перескажет то, что вы сказали, своими словами.

Психологический прием «Сказать, не говоря»

По себе знаю, что очень трудно говорить неприятное другому человеку. Допустим, вам надо сказать о намерении покинуть работодателя или о невозможности выполнить требования товарища. Как быть? Ведь просто ничего не сказать нельзя, но и рубить сплеча тоже не годится.

Самым простым решением в таких случаях может быть стратегия «Говорить, не говоря». Например, можно сказать: «Вы знаете, так складываются обстоятельства, что я, вероятно, в следующем месяце буду искать новый проект». Следует как можно больше говорить вводных слов, может быть, даже неверно ставить падежи и при этом наблюдать за реакцией собеседника: «О чем это он говорит? О каких-то обстоятельствах... Что, может быть, что-то там будет, а может, и не будет». Можно ли возражать в ответ на подобные фразы? Нет, поскольку вы вроде бы ни о чем конкретном не сказали. Но, с другой стороны, вы выразили намерение изменить ситуацию и развязываете себе руки для следующего хода, который может быть сделан через год, а может быть — через пять минут.

Первый пример использования приема «Говорить, не говоря». Исходная ситуация: вы желаете упорядочить визиты к клиенту и сократить количество срочных вызовов, но директор на это не соглашается.

Сражение № 1 (происходит в кабинете директора фирмы-заказчика).

Вы: А почему бы, уважаемый директор, нам не попробовать сократить количество срочных посещений с трех в неделю до одного?

Директор: Хм, я думаю, что мы пока не готовы к этому. Может быть, через три месяца?

Вы: Но пока будем исподволь готовиться?

Директор: Ну-у-у, не знаю...

Сражение № 2 (происходит за стенами кабинета директора среди сотрудников фирмы-заказчика).

Вы: Мы с директором решили попробовать сократить количество моих срочных посещений с трех до двух в неделю. Чтобы сделать каждое мое посещение более полезным для вас, мы сейчас заведем тетрадь, и в нее вы будете записывать вопросы к моему приезду.

Сотрудники: Ясно. Вот эта тетрадь подойдет?

В конце концов, не побегут же они к директору выяснять, был ли разговор про два посещения или нет, а если и побегут, то вы можете сказать директору: «Ну как же, помните, мы говорили о переходном периоде». Ведь фразу «через три месяца» можно понять как «давайте вернемся к этому делу потом», а можно и как «давайте сделаем так, чтобы через три месяца все было сделано».

Второй пример использования приема «Говорить, не говоря». Исходная ситуация: вы хотите перераспределить нагрузку между вами и вашим коллегой, с которым вы работаете над проектом, но он не проявляет к этому интереса.

Подготовка наступления:

Вы: Послушай, может быть, мы неправильно распределяем силы при выполнении данной задачи? Может быть, надо как-то реорганизовать нашу работу?

Собеседник: Что ты хочешь сказать?

Вы: Ничего конкретного. Просто мне кажется, что можно как-то оптимизировать нашу работу. Давай подумаем и через неделю обсудим это.

Реализация наступления (спустя неделю):

Вы: Мы в прошлый раз решили подумать о том, как оптимизировать нашу работу.

Собеседник: Я что-то ничего не придумал.

Вы: Я думаю, что можно сделать так и так.

Так устроена психика человека, что на предложение, высказанное ранее, можно в следующий раз опереться как на решенный аргумент. Часть слов забывается, остается общее впечатление. А общее впечатление можно интерпретировать как угодно (ну или почти как угодно).

Психологические приемы Владимира Владимировича Путина

У политиков есть чему поучиться. Например, после анализа выступлений Владимира Владимировича я взял на вооружение метод подстройки к собеседнику «по-путински». Сравните два варианта ответа на вопрос: «Почему столько насилия на экранах?»

- Потому что большинство тех, кто смотрит телевизор, по своему интеллектуальному уровню близки к уровню той муры, которую там показывают. Рекламные специалисты это давно поняли и скармливают с рекламой то, что ближе всего народным массам.
- Я давно задумывался над этим вопросом. Законодатели уже приняли ряд законов по этому поводу. Тема насилия связана и с воспитанием подрастающего поколения. К сожалению, простых путей решения этого вопроса нет. Нельзя взять и запретить... На мой взгляд, действительно много насилия. Однако хотел бы сказать, что рекламодателям, а именно они содержат телевидение, выгодно показывать рекламу в тех фильмах, которые смотрят больше людей. Следует менять структуру общества, и тогда наметятся определенные сдвиги.

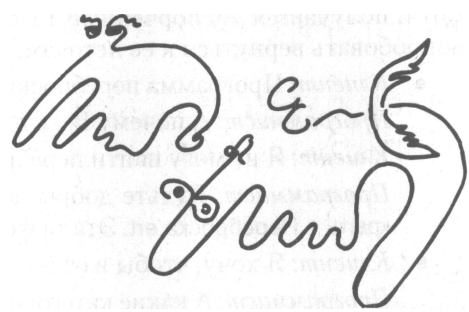
Какой вариант ответа вам кажется более подходящим для Президента? Я бы выбрал второй.

Принцип ответов Президента (как я его понял, и именно это я называю «методом подстройки») сводится к следующей схеме:

- Поздороваться, назвав собеседника по имени.
- Подстроиться под вопрос, сказав несколько предложений на затронутую тему.
- Ответить (или не ответить) на вопрос.
- Сделать выводы.

Как можно применить этот метод на практике, в бизнесе? Например, на вопрос: «Скажите, вы можете восстановить пропавшие данные?» — можно просто ответить: «Нет», а можно ответить: «Ваш вопрос не новый. Мне уже задавали его и в "МашРесурсе", и в "КомРесурсе", и в "КомАспекте". Я консультировался по проблеме восстановления винчестера, а именно на нем находится нужная вам информация. К сожалению, по заключению экспертов, вероятность восстановления информации очень мала. Я мог бы вам предложить страховочный вариант: регулярно архивировать информацию на других носителях, например на компакт-дисках». Вы напрямую не ответили на вопрос, однако у клиента не возникнет сомнений относительно вашей компетентности.

Прием «Ищи главного»



«Ищи главного». Общайтесь с главными, избегайте второстепенных. Главного можно отличить от неглавного по следующим признакам:

- Главный обладает административной властью: может вызвать на ковер, приказать сделать сверхурочную или дополнительную работу, принять волевое решение.
- Главный обладает «видением», может объяснить суть происходящего на предприятии.
- Главный принимает решение о выплате денег.

В случае возникновения проблем «главный» — основной помощник программиста. Нужно сразу обращаться к нему и просить помощи. Если вы работаете в команде, то главным может быть ваш начальник или старший товарищ.

«Слушай только компетентного специалиста». Если у вас возникает подозрение, что собеседник не тот, за кого себя выдает (например, не может вразумительно ответить ни на один ваш вопрос), попросите руководство заменить его или поищите замену сами. У некомпетентного человека бардак в голове. Он может двадцать раз в день менять свое решение, неожиданно вспоминать о новых подробностях или забывать о ваших договоренностях. Ответственность за реализацию бреда, который предложил некомпетентный человек, будете нести вы.

Кто-нибудь да должен знать ответ на интересующий вас вопрос. Возможно, ответ кроется совсем не в особенностях работы системы, а на теоретическом уровне.

Не нужно строить из себя начальника (даже если вы и есть начальник). Не все проблемы вы можете решить самостоятельно, и не все находится в вашей компетенции. В конечном счете, деньги программисту платятся за то, что все работает, а не за преодоление трудностей и героическое решение задач исключительно своими силами.

Искажение реальности

Так устроен мир, что любое отражение объекта ведет к искажению информации об *объекте*. Если мы смотрим на снимок объекта в фас, то скрытым остается то, что находится сзади и сбоку. Искажение информации при ее передаче — обычное дело.

Происходят подобные процессы и при общении с клиентами. Когда человек обдумывает проблему, он пропускает информацию через фильтры своего сознания и искажает ее. Потом он излагает свое видение проблемы, а вы слушаете и пропускаете информацию через свои фильтры и снова ее искажаете. Вот и получается «испорченный телефон». Как понять суть проблемы? Следует попробовать вернуться к ее истокам. Приведем несколько примеров.

- *Клиент:* Программа переброски не работает.

  Программист: А почему Вы считаете, что программа не работает?

  Клиент: Я не могу найти переброску среди отчетов.

  Программист: Будьте добры, выполните следующие действия: Файл • Открыть • Переброска.ert. Эта та форма, которую вы потеряли.
- *Клиент:* Я хочу, чтобы в отчете выдавались цены категории А.

  Программист: А какие категории используются, и почему следует выдавать цены категории А?
- *Клиент:* Сделайте в конце печатной формы несколько пустых строк.

  Программист: А зачем Вам нужны пустые строки?

  Клиент: Оператор будет записывать имя экспедитора, который должен отвозить груз.
- *Клиент:* Сделайте учет кег (возвратной тары) по номерам, которые выбиты на кегах.

  Программист: А для чего это нужно?

  Клиент: Так делают все.

  Программист: А что заставило других владельцев кег вести учет по номерам?

  Клиент: Когда экспедитор приезжает к клиенту, он не может определить, какая из пустых кег принадлежит предприятию, а какая не принадлежит.

\

Проблемы, которые мы себе выбираем

Рассмотрим группу специалистов. Один член группы работает только на передовой (хлебом не корми — дай нового клиента обработать). Другой интересуется только проблемами железа. Третий занимается только разовыми заказами. Казалось бы, в результате внутригруппового взаимодействия навыки наиболее продвинутого товарища (у которого доходы существенно превышают доходы остальных) должны передаваться другим членам группы. Но этого не происходит: железячники так и остаются железячниками, сетевики — сетевиками, программисты средней руки не идут вверх. Почему?

Я думаю, что причиной может быть психологическая нацеленность на тот вид проблем, которые специалист настроился решать. Если «продвинутому» члену группы будут задавать «железные» вопросы, то он отфутболит вопрошающего к «спецу», но если будет нужно урегулировать конфликт с заказчиком, то он схватится за возможность подраться, как черт за сухую грушу. И дело не в том, что один член группы — руководитель, а другой — подчиненный или что один может переустановить сервер, а другой — нет; такие группы связывают неформальные попарные связи, которые строятся на личных взаимных интересах. Причина отчасти кроется в психологических принципах: «Если не я, то кто?» или «Начальник сказал — надо, значит, следует не раздумывая бросаться в бой».

Следует чаще задумываться, стоит ли решать проблему или, может быть, лучше отдать ее на растерзание коллегам, или вообще превратить в заказ, который продать сторонней фирме.

Кстати, попробуйте проанализировать ограничения, которые построили для себя ваши окружающие, а после этого определите проблемы, которыми предпочитаете и избегаете заниматься вы.

Проигрывай — это полезно!

У каждого специалиста вырабатывается свой подход к решению профессиональных проблем. Одного не убедишь в том, что TCP/IP хуже, чем NetBEUI, другого — что один стиль программирования лучше, чем другой. Специалист знает, как быстро решить задачу из свой области. Он может на автопилоте преодолевать сложные и запутанные места. Он работает с большой скоростью, и результаты работы вполне устраивают клиентов, которые привыкли встречать менее опытных или менее красноречивых спецов.

Мастерство специалиста по-настоящему подвергается сомнению тогда, когда он сталкивается с таким же по величине или более крутым специалистом. Тогда вдруг оказывается, что вы пользуетесь устаревшей технологией или не таким голосом поете клиенту, и вообще, «Торговля» версии 7.7, на которой вы построили свою карьеру, — просто игрушки по сравнению с «Торговлей» 8.0.

Важно встречать тех, кому можешь проиграть! Проиграл — значит, теперь ты знаешь, в какой области надо совершенствоваться: в общении, в технологии, в психологии... А какой толк от того, что выиграл? Выигрыши не обогащают, они лишь сеют семена гордыни.

Так что ищите тех, кому вы можете проиграть!

Как повысить свою значимость

Самый простой способ повысить свою значимость кроется в буквальном значении слова «повысить». Сядьте на стул повыше и расправьте позвоночник. Я знаю, как это тяжело, но, поверьте, дело того стоит. Зачем сидеть прямо, хорошо написал Норбеков: «Человек — это физическое тело, мысли и эмоции. Чтобы мысль работала, как мысль победителя, следует придать телу вид победителя, а эмоциям — хорошее настроение».

Второй шаг повышения значимости себя — это самопрограммирование на успех. Например, можно написать на листочке фразу: «Почему этот год будет для меня годом успеха и процветания?» — и повесить листок рядом с рабочим местом. Каждый раз, когда взгляд будет пробегать по этой фразе, мозг будет запускать программу по поиску лучшего решения. Вы спросите — лучшего решения чего? Того, что находится в вашем подсознании.

Выращивание амбициозных планов

Самый простой способ вырастить и выполнить амбициозные планы — это общаться с амбициозными людьми. То, что смог сделать один человек, другой тоже сможет сделать. Успешные люди похожи на нас с вами: одна голова, две руки, два глаза. Но чем они отличаются от нас? Общаясь с победителями, следует всматриваться в то, как они работают, и копировать их стратегии.

Возможно, вы возразите, что вы сидите в болоте и вокруг вас одни лягушки. Тогда возьмите в качестве примера Сороса, Гейтса, Нуралиева или Путина.

Непсихологические приемы борьбы с хамами

Хамство иногда незаметно. Клиент может вдруг начать покрикивать или придираться к мелочам. Типа, почему это ты ходишь в клетчатом пиджаке. Отвечать на такие претензии невозможно. Если начнешь хамить — победил хам. Начнешь отвечать, почему ты ходишь в таком пиджаке, — хам тоже победит. Один из способов борьбы с хамами — это попытка поставить перед хамом «буфер».

Допустим, начальник отдела выходит из себя по поводу, совершенно не связанному с работой программиста. Вы можете попросить более высокое руководство повлиять на начальника отдела так, чтобы тот передавал вам свои претензии (сомнения, негодования) не непосредственно, а через своего подчиненного. В этом случае хам будет вынужден сформулировать проблему, а не вести беседу в стиле «сам дурак».

Нужна ли психология программисту?

Если вы считаете, что психология не для вас, а ваше дело — программировать, то дело как раз обстоит с точностью до наоборот. Человек, выбравший работу в об-

ласти 1С-программирования, не сможет обойтись без общения с клиентом. Наблюдая за работой многих технарей-программистов, да и сам являясь таковым, могу заключить, что очень многим специалистам не хватает именно навыка общения с клиентами, особенно — с руководством фирмы-заказчика. Не задумываясь, обращайтесь к начальникам, если вы чувствуете, что они могут придать работе ускорение. Другая психологическая проблема — неуверенность в себе. С ней вам поможет справиться аутотренинг и здоровый образ жизни.

Глава 7
Организация бизнеса

Эта глава содержит информацию об организации бизнеса в области 1С-программирования и практические упражнения по нахождению новых клиентов.

Для чего нужен свой бизнес?

Предприниматель является хозяином своих действий. Хочет — направляет корабль в сторону «1С», хочет — плывет в сторону «Паруса». Предприниматель — творец, причем творец, ответственный за плоды своих решений. Предприниматель — человек, которому для работы не нужны ни дипломы, ни сертификаты, ни покровительство хорошего дяди.

А может быть, сначала поработать на дядю?

Вы можете возразить, что перед самостоятельным движением вперед стоит набраться опыта. На это я бы возразил так: работая в чужом бизнесе, вы не приобретете уверенность. Опыт приобретете, а уверенности в том, что вы сможете идти один и вести за собой, — нет.

Работая в чужом бизнесе, вы будете развиваться однобоко. Допустим, в вашу компетенцию входит работа правой ногой, в то время как другой работник отвечает за левую ногу. Спрашивается, хороший ли опыт вы приобретете на такой должности? Да, вы отменно научитесь махать своей конечностью и, после того как уволитесь, может быть, устроитесь работать «правоножным» на более высокооплачиваемую работу. Но в деле собственного движения на двух ногах работа «на дядю» будет вредить. Кроме того, не ясно, сколько времени достаточно потратить на приобретение опыта и когда обучение следует заканчивать.

Приведу еще одно обобщение. Предприниматели вряд ли согласятся работать в одной упряжке. Они, как пауки в банке, будут бороться за самовыдвижение вперед. И если один (ваш работодатель) — паук, а другой (вы) — муха, то сотрудничество будет пагубно для вас.

По теме организации своего бизнеса я посоветовал бы посетить сайт http://moroz."nego.ru Юрия Мороза.

Сколько денег необходимо для начала дела?

1С-бизнес не требует больших финансовых вложений, в отличие от оптового, розничного, производственного бизнеса, которые требуют денег. Все, что нужно для дела, программист носит с собой (в черепной коробке) в виде знаний и связей с клиентами. Поэтому для организации дела нужны только уверенность в успехе и сотня визиток.

Нужно ли иметь специальное образование и сертификаты?

Для организации дела в области «1С» вам понадобятся:

- общие навыки программирования, а также навыки программирования в среде «1С»;
- знание бухгалтерского учета;
- знание практической психологии.

Рискну утверждать, что все это можно приобрести только самообразованием, и специальное образование здесь не помощник. С практической психологией все ясно: психологию невозможно изучать в лекционном зале — только «в полях». Бухгалтерский учет не столь мудрая наука, чтобы на нее тратить пять лет: достаточно трех-четырех книжек и вдумчивого чтения текстов законов. Что же касается программирования, то, для того чтобы считать себя специалистом, необходимо выполнить пару проектов, причем полностью — от разработки до внедрения.

Если речь идет о сертификатах, то для работы с «1С» требуется три сертификата: в области программирования, в области бухгалтерского учета и в области практической психологии, — а не один (сертификат 1С-программиста), как обычно требуют при приеме на работу.

На вопросы заказчиков о наличии сертификатов при их отсутствии я бы советовал переводить разговор на обсуждение проблем клиента.

Работники, офис, организация бизнеса

Существует мнение, что организация бизнеса обязательно связана с наймом работников, арендой офиса и содержанием секретаря и штата бухгалтеров. Современные технологии меняют ситуацию. Попробую убедить вас, что каждый из этих атрибутов бизнеса не является необходимым.

Ориентация на наемных работников пагубна, особенно в таком интеллектуальноемком деле, как программирование бухгалтерских систем. Работник норовит украсть, побольше узнать и сработать «налево». Нормировать труд программиста невозможно: слишком не похожи друг на друга задачи. Поэтому правильнее строить работу не с работниками, а с партнерами, когда каждый просматривает последствия своей работы.

Всеобщая телефонизация, в эпоху которой мы живем, ставит под сомнение пользу от работы секретарей. Мы сами себе ходячие офисы, готовые на ходу принимать решения и координировать свою работу с такими же ходячими офисами, как мы сами.

Что касается бухгалтерии, то ее можно отдать специализированным фирмам, и лучше на начальном этапе вести учет от лица другой фирмы. Следует помнить, что регистрация предприятия будет требовать от вас ежеквартальной сдачи декларации по НДС и ежегодных бумаг в пенсионный фонд и другие фискальные органы, а также затрат на обслуживание расчетного счета в банке и покупку кассового аппарата. Затраты на ведение бухгалтерского учета можно сократить, если перевести учет на упрощенный метод начисления налогов.

Место встречи с товарищами предприниматель-новичок может назначать у клиентов, в автомобиле, в кафе, дома или в Интернете. За без малого шесть лет сотрудничества с почти сотней клиентов ни один из них не сподобился побывать у нас в офисе. Наверное, никто из них и не знает, что офиса у нашей программистской группы просто нет.

Правда, следует оговориться, что постепенно с ростом дела необходимость в офисе все же возникает, поскольку начинает расти специализация членов группы и приходится согласовать между собой их действия. Такое согласование может быть достигнуто только через одновременную встречу многих людей, а не через попарные контакты.

Возможные типы бизнеса в сфере «1С»

Из всего многообразия возможных видов предпринимательской деятельности, связанной с компьютерами и «1С», я бы выделил следующие:

- коробочные продажи,
- работа с оплатой по часовому тарифу,
- договора на абонентное обслуживание,
- совмещение программного обслуживания с обслуживанием компьютеров,
- обучение.

Коробочные продажи

Оставлю «коробочный бизнес» на долю крупных контор, поскольку при продаже «коробки» начинающий программист должен будет обеспечить и внедренческое обслуживание, что явно невозможно для новичка.

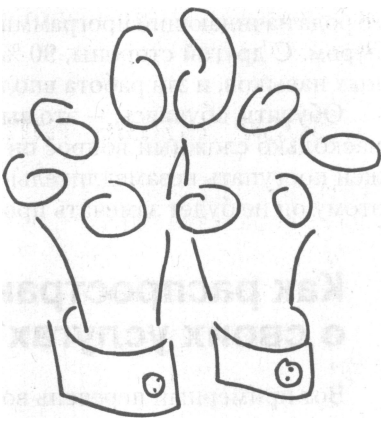
Работа с оплатой по часовому тарифу (повременная работа)

Видимо, повременный бизнес тоже следует считать прерогативой крупных фирм. Новичку приходится браться за любую подвернувшуюся работу и подходить к каждому заказу индивидуально. Со временем появляется постоянный круг клиентов, а также сотрудников (товарищей), и лишь тогда появляется возможность перехода на повременную работу.

Например, когда я начал заниматься собственным бизнесом, то испытывал недостаток в клиентах и старался каждого клиента перевести на фиксированную оплату в месяц. Систему убеждения я строил на том, что при повременной системе оплаты программист не может гарантировать комплексное обслуживание программы. Ведь программист дает гарантию только на свои работы. Если в следующий раз приходит другой программист, то он как бы обнуляет результаты работы своих предшественников. Клиенты, которые соглашались на работу со мной, перед этим много раз обжигались на повременщиках, которые «срубали бабки» и исчезали.

Теперь я вижу, что переход на повременную оплату по прошествии нескольких лет после начала дела позволяет легко распределять работу между специалистами. В договорах на фиксированную оплату в месяц один специалист вынужден делать все работы, поскольку трудно определить, как следует делить будущий пирог. В повременной оплате такие проблемы отсутствуют, поскольку сумма оплаты оговаривается сразу.

Абонентское обслуживание



Под абонентским обслуживанием я понимаю договор на постоянную сумму, которую клиент платит раз в месяц.

За плату, равную 30-50 % от оплаты специалиста в режиме постоянной работы, клиент получает возможность в любой момент получать телефонные консультации и может рассчитывать на плановые посещения специалиста. Поскольку финансовый вопрос решается раз в год при определении суммы ежемесячного обслуживания, то проблема с оплатой решается гораздо легче, чем в случае повременки.

Работа в режиме абонентского обслуживания предпочтительна для новичков. Представьте, что, после того как программист сделал работу, обнаружилась ошибка, связанная с неверной постановкой задачи клиентом. Как быть? Брать дополнительную плату и доказывать, что клиент неверно объяснил задачу, или выполнить доработку за свой счет? При этом следует иметь в виду, что технические задания составляются в лучшем случае на огрызке черновика, а не в виде законченного договора. Бывалый программист, скорее всего, избежит такой ситуации, однако новичок-предприниматель не имеет опыта словесных баталий, поэтому вариант фиксированной оплаты в месяц для новичка наиболее приемлем.

ПРИМЕЧАНИЕ Пример договора на абонентское обслуживание вы можете найти в Приложении Б. Одним из условий договора является ежемесячная оплата определенного количества часов работы программиста. Прошу обратить внимание на пункт договора, в котором говорится, что сумма оплаты остается неизменной, если нагрузка оказывается больше или меньше запланированной на 30 %.

Совмещение компьютерного обслуживания с обслуживанием 1С-программ

Видимо, это наиболее перспективный бизнес для новичков. Можете приплюсовать сюда услуги заправки картриджей, и кусок хлеба у вас будет всегда. Однако с другой стороны, обслуживание компьютеров связано в том числе и с настройкой сетей, перезагрузкой серверов и борьбой с вирусами, а это значит, что вы обрекаете себя на ночную работу. Заказы на переустановку сервера могут дорого стоить, но требуют много времени, поэтому не каждый крутой специалист согласится обменять время на деньги, полученные с таких работ, оставляя эти работы новичкам.

Обучающий бизнес

Я считаю идеальным такой процесс обучения, когда программист-новичок обучает с нуля пользователя-новичка. В этом случае можно оценить коммуникабельность, психологическую устойчивость и познавательные способности обучающего. У ученика возникают два вида затруднений: затруднения в усвоении нового материала и затруднения механического закрепления навыков. Проблемы первого рода начинающий программист может решить, консультируясь с опытным партнером. С другой стороны, 90 % обучения — это помощь в закреплении полученных навыков, и эта работа вполне по плечу программисту.

Обучать, обучаясь, — это высший пилотаж. Новичок-ученик обычно не знает, насколько сложный вопрос он задает, и ему не известно, на какие вопросы должен поступать незамедлительный ответ, а с какими надо разбираться долго. Поэтому он не будет замечать пробелов в знаниях обучающего.

Как распространять информацию о своих услугах?

Вот примерный перечень возможных маркетинговых шагов по созданию своей сети клиентов:

- реклама в газетах,
- создание сайта,
- продвижение услуг через Знакомых,
- обход возможных клиентов с проспектами и раздаточными материалами.

Первое, что всегда приходит на ум, это дать рекламу в печатных изданиях. На самом деле этот способ является самым неэффективным для новичка. Ну не читают 1С-клиенты газет! Только постоянная реклама в специальных печатных изданиях может дать результат, а такую стратегию, требующую больших финансовых вложений, может позволить себе только крупная фирма, а не начинающий частник. Однако вы можете попробовать бесплатную разновидность такой рекламы, напечатав объявления в газетах бесплатных объявлений.

На сайт, на мой взгляд, тоже не следует возлагать особых надежд. Я создавал свой сайт (www.prof.narod.ru) много лет назад, и до сих пор он не дал ни одного

клиента. Причина отсутствия притока клиентов в том, что клиенты не ходят по Интернету; они ходят по земле и сидят в конкретных зданиях, а не в гиперпространстве.

Следует быть там, где клиенты концентрируются, и сделать так, чтобы защитный барьер, который всегда существует при первом знакомстве, был максимально низким. Иначе вашу работу можно будет сравнить с выращиванием помидоров в пустыне или с заигрыванием хипповато одетого молодца с интеллигентной девицей.

Самым лучшим и самым безотказным, на мой взгляд, способом организации сети клиентов является реклама «из уст в уста». За свой многолетний опыт работы в качестве программиста я ни разу не давал объявления в газетах и журналах.

Следует для начала просто обойти всех своих знакомых и, рассказав им, чем вы собираетесь заниматься, попросить назвать имена тех, кто мог бы заинтересоваться вашими услугами. После этого надо лично встретится с каждым порекомендованным и рассказать ему о том, чем вы занимаетесь. Это процесс длительный, но действенный.

Еще одним способом улавливания клиентов является самостоятельный обход предполагаемых мест их скоплений. Этот способ наиболее труден для программистов, привыкших общаться с железом, а не с людьми. Но ничего не поделаешь: предприниматель — это не только программист, но и актер. А это значит, вы должны понимать, в какой момент времени на вашем лице должна быть улыбка, а в какой — выражение сосредоточенного внимания. Вы должны знать, что нужно говорить, а что не нужно. Вы должны научиться мгновенно парировать вопросы, касающиеся всевозможных сертификатов, гарвардских дипломов и крутых систем конкурентов. И главное — вы должны научиться нравиться людям.

Актерское мастерство вырабатывается самым обыденным способом — беседами с возможными клиентами. В этом деле вам никто не поможет: ни тренинги, ни советы, ни книжки, ни обучающие компакт-диски, ни советы друга. Чем раньше вы начнете пробовать силы в практике прямого общения, тем больший успех вас ждет.

Самым мощным обучающим моментом является ваша попытка договориться с клиентом. Ваш собеседник иногда может вообще вас не слушать, или задавать вопросы невпопад, или просто показать пальцем на дверь. Однако после сотни-двух встреч вы начнете понимать, что к чему в этом мире и как на самом деле нужно говорить.

Строительство клиентской сети

Под *клиентской сетью* я понимаю множество клиентов, которые с определенной частотой обращаются к вашим услугам.

Целью любого бизнеса является создание клиентской сети. Не каждая сеть подойдет вам, а только сеть, созданная именно под вас. Для развития дела вы последовательно должны решать следующие вопросы:

- Как мне найти новых (выгодных) клиентов?
- Какие клиенты мне невыгодны?
- Как их отсеять?

Первая проблема понятна, а последняя может вызвать удивление. Зачем отсеивать клиентов, если мы строим сеть? Дэвид Паккард (основатель компании Hewlett Packard) утверждал, что смерть в бизнесе чаще всего наступает не от голода, а от несварения. В правильно поставленном деле клиенты будут постоянно прибывать, и вам важно не набрать клиентов больше, чем вы сможете обслужить.

Кстати, отсеивать лишних разовых клиентов вы можете, предложив им заключить договор на абонентское обслуживание.

Меняй свое дело на 30 % каждые три месяца

Развитие — это движение. Движение — это изменение. Чтобы сделать развитие максимально быстрым, следует прилагать усилия к изменению.

Вы можете легко определить, что следует изменить в своем деле, если зададите себе простой вопрос: какая работа занимает большую часть вашего времени и приносит меньше всего доходов? Как только вы определились с пиявкой, высасывающей из вас соки, принимайте меры по ее устранению. Изменения не всегда связаны с гигантоманией — с увеличением количества работников или клиентов. Изменения могут быть связаны и с сокращением.

Может быть, в результате своего анализа вы придете к выводу, что лишним для вас является слишком дорогой офис, а может быть — непонятливый работник или излишне требовательный клиент, или вы решите, что убыточным является сам бизнес.

К сожалению, этот совет можно отнести только к предпринимателям, то есть к людям, которые вольны в выборе своего пути. Работники же делают то, что им велено, и не могут позволить себе роскошь задаваться вопросом, какой тупой работой они занимаются. Вернее, вопрос этот наемные работники задать могут, а вот ответ, скорее всего, приведет их к тому, что для избавления от тупой работы следует вести бизнес самостоятельно.

Как отстреливать клиентов?

Бизнес — это поиск тех клиентов, которые нужны, и освобождение от тех, которые не нужны. Силы человека ограничены, он не может тянуть тележку и еще маленький воз. Найти такую работу, чтобы быть занятым по уши и получать гроши, — большого ума не требуется. А вот изменять ситуацию так, чтобы получать наибольшие выгоды при наименьших затратах, — это искусство.

Проходит время, информация о вас передается из уст в уста, постепенно появляются более привлекательные клиенты. Раньше вы брались за любую работу, от восстановления «винды» и обучения до прокладки локальной сети. Теперь можно остановиться и задать себе вопрос: кто из вашей компании самый проблемный и наименее денежный, то есть чем и кем можно пожертвовать, чтобы сделать шаг вперед?

Возможно, вас гложет страх (от добра добра не ищут, и вообще, лучше синица в руках), но если нет изменения, то не будет и развития. Изменения важны, хоть хорошие, хоть плохие.

Как сказать, что вы хотите уйти?

По себе знаю, что говорить неприятные вещи тяжело, особенно тем, с кем давно сотрудничал. В этом случае может помочь психологический прием намека.

Сравните следующие фразы:

- Я нашел более выгодного клиента, поэтому вы меня больше не устраиваете. Ищите за те гроши, что вы мне платите, другого дурака.
- Через месяц я прекращаю работу с вашим предприятием.
- К сожалению, у меня возникла ситуация, которая привела меня к перегрузу. Дело в том, что я перестал справляться со всеми своими клиентами. Вполне возможно, что вам придется искать нового программиста.

Очевидно, что последний вариант предпочтителен. Здесь много вводных слов, конкретно ничего не говорится (может быть, будет, а может, и нет), но, используя выражения, подобные этому, вы можете в любой момент сделать откат или начать более решительные действия.

Уходить, не уходя

Итак, вы идете вперед, а за вами возникает вакуум, который надо заполнить. Это необходимо для того, чтобы не терять прошлые контакты, чтобы о вас складывалось благоприятное впечатление, чтобы иметь шанс вернуться (хотя это вряд ли произойдет), чтобы люди, которые работают у ваших клиентов, перейдя в новые фирмы, вспомнили о вас.

Возникает проблема: как уйти, но остаться? Решения есть. Вы можете передать клиента фирме схожего направления (получив 5-10 % от суммы нового договора) или внедрить к клиенту своего ученика.

О ценах

Никогда при первой встрече не называйте цены. Тому есть несколько причин.

Во-первых, трудоемкость решения будет известна только после более близкого знакомства с задачей. Правильно оценить конкретную услугу при первой же встрече нельзя. Первое впечатление обманчиво, и даже когда кажется, что все ясно, как божий день, очень вероятно, что впереди вас ждут сложности, о которых вы и не подозревали. Опыт показывает, что первая оценка оказывается занижена где-то в три раза.

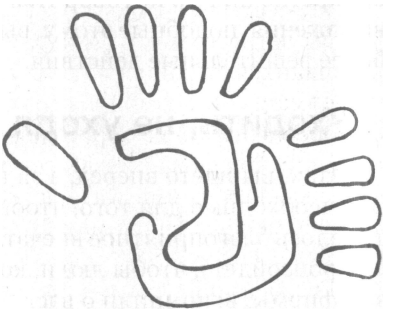
Во-вторых, частенько при прочих равных факторах, клиенты соглашаются на разные договоры. Один платит повременно, с другим можно заключить договор на конкретную сумму с ежемесячными выплатами. Крупное предприятие готово платить больше, но за работу попросит гарантийное и постоянное обслуживание. Мелкое предприятие заплатит меньше, зато заказ небольшого предприятия может выполнить менее квалифицированный специалист.

Кроме того, клиент может ошибаться в постановке задачи, может неправильно вас понять, наконец, он может просто собирать информацию. Если в первый раз вы говорите с клиентом по телефону, то лучше всего назначить встречу и обсудить проблему с глазу на глаз. При встрече можно определить «ценность»

клиента. Например, если клиент находится рядом с вашим офисом, вы можете предоставить ему особые выгодные условия, а если он расположен далеко, заломить двойную цену.

И все же, если клиент упорно просит назвать цены, можно ответить, что стоимость ваших услуг не сильно отличается от среднерыночной. При этом следует сделать акцент на особенности ваших услуг (например, сказать, что вы работаете в определенном районе или предпочитаете заключать договоры на постоянное обслуживание). Иными словами, на конкретный вопрос о ценах вы отвечать не должны.

Проведи 200 встреч



Одним из маркетинговых шагов в области нахождения новых клиентов для вашей сети может быть проведение переговоров с потенциальными клиентами. Стоит оговориться, что речь идет о выборе новых клиентов, а не о тех, кто выбирал вас. Вполне возможно, что вы в течение дня отвечаете на множество телефонных звонков и встречаетесь со многими людьми — но эти встречи не в счет. Это были люди, которые встретились с вами, чтобы оценить, стоит ли иметь с вами дело. Вам нужно самому выбирать, с кем сотрудничать.

Выберите район города, на ваш взгляд, наиболее подходящий для обхода. Это может быть район промышленных предприятий, оптовых баз или офисных зданий. Далее следует обойти каждую фирму района и рассказать ей о своих услугах. Желательно иметь при себе визитки и проспекты.

Тренинг: напиши проспект о своем деле

Предприниматель — это не всегда интеллектуал. Предприниматель — это, скорее, человек, который точно знает, чего хочет, знает технологию зарабатывания денег и не дергается от одного проекта к другому. Предпринимательство также связано с общением с людьми и умением убеждать.

Я предлагаю вам попробовать выполнить тренинг на формулирование идеи предпринимательства и на развитие навыков общения.

Подумайте, что вы можете делать из того, за что другие могли бы заплатить деньги. Опишите свою услугу так, чтобы получился проспект, который можно распечатать на принтере. Напечатайте 10 проспектов и раздайте их. По реакции получивших ваш листок людей внесите изменения в проспект. После этого напечатайте еще 100 экземпляров проспекта и раздайте их. Все до единого.

Мой опыт показывает, что розданные проспекты и буклеты совсем не обязательно отправляются в корзину для бумаг, как только вручивший их скрывается за дверью. Как оказалось, иногда проспект может оставаться на столе получателя больше года, причем даже если он не представляет большой художественной цен-

ности (мой, например, был напечатан на белой бумаге с помощью обыкновенного черно-белого ксерокса).

Выполнение упражнения должно хорошо напрячь мозги. Если вы ни разу не писали рекламных объявлений, это будет хорошим поводом для анализа того, как другие справлялись с подобной проблемой. С другой стороны, вам будет необходимо общаться с потенциальными клиентами и потребителями ваших услуг, и данное упражнение даст вам бесценный опыт общения.

ПРИМЕЧАНИЕ Пример проспекта, с которого я начинал предпринимательскую деятельность, можно найти в Приложении Г.

Расскажи о себе всем

Из ста встреч с клиентами удачей завершаются только пять встреч. Представьте, что вы встретились с *сотней* человек, а результат — с гулькин нос. Как сделать купившихся на ваши посулы людей потенциальными распространителями вашей информации и возможными будущими покупателями? Ответ один: произвести на собеседника благоприятное впечатление. Согласно популярным продажным теориям, поддержание благоприятного впечатления о своем продукте — задача более важная, чем непосредственная продажа. После того как создана благоприятная атмосфера и сломан лед недоверия к продукту, продажи происходят сами собой.

Из своего опыта я могу заключить, что 80 % моих теперешних клиентов было заработано в результате рекомендаций. Каждому своему знакомому я говорю, что я программист и работаю в области обслуживания бухгалтерского программного обеспечения. Мои знакомые запоминают это и при удобном случае рассказывают обо мне другим.

Мне нравится программировать. Мне нравится быть человеком, свободным от «дурака-начальника». Я уверен, что без подобной стратегии саморекламы у меня не получилось бы дела.

Единственная возможность самореализоваться — это делать любимое дело, и с этим мало кто будет спорить. Наладив поток потенциальных заказов, вы сможете выбирать заказы по своим силам, получите возможность экспериментировать и обретете перспективу роста в выбранном вами (а не начальником) русле.

Даже если вы не ведете никакого своего дела, это не значит, что вы не можете следовать вышеуказанному принципу «Расскажи о себе всем».

Напутственное слово агенту

Не начинайте беседу до тех пор, пока не решите для себя, какова ее цель: вручить визитки, выяснить позицию руководителя, обновить информацию о себе. Цель встречи необязательно должна заключаться в продаже продукта. Лучше подменить цель «заключить долгосрочный договор» или «срубить 500 рублей» на другую — «вручение рекламных материалов».

Замените слово «продажа» на любое другое по своему вкусу! Скажите, например: «Я решаю проблемы почти бесплатно».

Следующий важный пункт — настроить себя на косвенный результат. Опыт показывает, что лишь пять встреч из ста заканчиваются удачно, поэтому если вы поставите конечную цель — «деньги», то независимо от финансовых результатов вы все равно будете разочарованы. Результаты вашего обхода обязательно будут, но не сейчас, а через некоторое время. Но если вы поставите цель «познакомиться с сотней проблем предприятий района», то но завершении акции вы будете чувствовать себя более комфортно, чем в случае концентрации на деньгах.

Прокрутите в голове беседу. Прокрутка беседы в голове — одна из составляющих самоубеждения. Говорят, был один баскетбольный тренер, который на тренировке сажал игроков в круг и заставлял мысленно совершать броски и передачи. После таких занятий команда играла так же хорошо, как после обычных тренировок. В реальной жизни я использую такой подход перед программированием. Сначала я сажусь и представляю, как программа (отрезок программы) будет работать, потом представляю, как я буду программировать. Постепенно возникает непреодолимое желание воплотить идею, и я начинаю писать код.

Придумайте *фразу самовозгонки*. Например, у меня она такая: «Вижу цель! Ломаю преграды!» Фраза самовозгонки позволяет выйти из сонного состояния и настроиться на победу. Бывает, что многие дела, к сожалению, так и не начинаются, потому что, пока раскачиваешься, наступает время переходить к другой работе. Фразы самовозгонки позволяют сократить период раскачки.

Подготовьте семь простых вопросов, на которые клиент может ответить только «да». Например: «Есть ли у вас сейчас пара минут для короткого разговора?», «Используете ли вы 1С-программы?», «Хотите ли вы решить какие-либо вопросы по «1С:Бухгалтерии» прямо сейчас?», — и так далее, в зависимости от конкретной ситуации. Если вопросы составлены удачно, то клиент, ответив несколько раз кряду «да», уже не в состоянии сказать «нет». Коллекционируйте такие вопросы, заимствуйте их у клиентов и конкурентов: эта коллекция — большая ценность!

Напишите речь *«Семь причин, по которым вы должны воспользоваться нашим предложением»*. В Интернете есть сайт Владимира Шахиджаняна «Учись говорить публично» (http://1001.vdv.ru/books/speak). Одно из его упражнений заключается в том, что следует придумать речь на любую тему. Отрабатывая речь, проговаривайте ее перед разными аудиториями, в том числе и перед теми, кто вас не собирается слушать или считает вашу речь полной ерундой. Очень эффективно записать и просмотреть свою тренировочную речь на видео.

Запоминайте возражения. Возражение — это скрытое пожелание, которое можно обратить на свою сторону. Приведу пример из практики. Клиент говорит: «Пожалуй, приходящий компьютерщик нас не устроит. Объяснишь одному, что нужно делать, потом появляется другой, и объяснения приходится повторять заново. Наверное, нам нужен постоянный компьютерщик, ведь у нас солидное предприятие». Вы продолжаете его мысль и говорите: «Мы и есть постоянные компьютерщики. Мы заключаем договоры именно таким образом, чтобы каждый клиент имел гарантию качественного обслуживания».

Можно также запоминать те моменты беседы, когда клиент проявил самый высокий эмоциональный всплеск: улыбнулся шире обычного, повысил голос, стукнул кулаком по столу, взмахнул рукой. Позже можно сделать анализ беседы и определить, что же его в этот момент так всколыхнуло.

Узнавайте всю информацию о клиенте. Если ваш товарищ уже был у клиента, то следует выяснить у него, к кому следует обращаться, каких вопросов следует избегать. Запишите имена, фамилии и телефоны! Когда Путин говорит со страной, он всегда называет человека по имени. Путин знает, что имя человека — не пустяки!

Подбирайте методы воздействия на подсознание собеседника. Метафоры, сравнения, обобщения, поговорки, анекдоты, байки — все это позволяет воздействовать на человека на подсознательном уровне. Используйте рисунки, графики, чертежи. Например, на вопрос: «Почему компьютеры ломаются?» — можно ответить так: «Конь на четырех ногах — и тот спотыкается, а куда уж компьютеру — делу рук человеческих?»

Постарайтесь *вручать материалы в руки.* По крайней мере, визитку вручайте только в руки. Объясните, что на ней написано, чтобы человек обратил на это внимание. Если он говорит, что у него уже есть ваша визитка (это хорошо! он помнит о вас!), следует сказать, что это новый вариант, что в него внесли поправки, и вручить еще одну визитку. Пусть ваших визиток будет много, тогда они будут закрывать визитки ваших конкурентов. Мастера продаж на улицах первым делом вручают товар в руки будущему покупателю, поскольку так будет проще убедить его купить товар. Продавцы знают, что делают!

Репетируйте речь перед встречей с клиентом. Обдумайте возможные возражения клиента. Опыт показывает, что спланированная акция всегда более успешна, чем экспромт. Правда, опыт проведения репетиций показывает, что реальная встреча идет совершенно другим путем, чем было предусмотрено, но это не значит, что ход разговора не следует планировать. Лучше всего отрепетировать выступление в кругу «соратников», которые подкинут новые идеи или подскажут, какие фразы следует построить по-другому.

Не доверяйте тому, что говорит клиент. Я говорю о недоверии в философском смысле: подвергните все сомнению и тщательному пересмотру. Попросите клиента объяснить проблему в более широких категориях. Решение проблемы может лежать совершенно в другой области, чем кажется заказчику. Также важно узнать предысторию проблемы, тогда вы сможете оценить потенциал будущих затруднений и «денежную мощность» клиента (стоит ли его обхаживать, если дело стоит копейки).

Никогда не говорите: «Не знаю». Лучше скажите: «Разберусь и перезвоню» или «Вы знаете, в мою задачу сегодня входит посещение клиентов в другом районе, давайте я к вам направлю своего товарища, он разберется в вашей проблеме». Избегайте употреблять фразы: «Не сделал», «Не успел», «Не смог». Говорите: «Случилась непредвиденная ситуация, которая привела к временной неработоспособности».

Студенты — материал для экспериментов

При социализме существовали пионерская организация и комсомол. Честолюбивый молодой человек, решивший занять руководящее место в социалистическом обществе, мог начать свою карьеру с этих организаций. Постепенно он учился выступать, говорить «правильные слова», организовывать собрания,

«руководить». Жившие во времена этих организаций могут возразить, что в них была только форма, а содержания не было. Но лучше плохо, чем никак.

Вступая на путь предпринимательства, вы неизбежно столкнетесь с необходимостью расширять фронт работ за счет наемных работников или партнеров. А как руководить, если вы никогда не руководили? Вам нужен будет такой материал, чтобы за меньшие деньги получить максимум опыта. Таким материалом являются студенты. Студенты еще не искушены в завуалировании своих интересов, их поведение легче вычислить и предсказать, чем поведение опытного работника. Они с большей готовностью будут подчиняться, чем взрослый человек.

С первых моментов работы со студентами, правда, начнутся и проблемы. Вот некоторые реплики тех, кто в первый раз начал руководить:

- Я не думал, что он этого не знает.
- Ему, оказывается, нужно показывать, где какие дырки сверлить. Без этого он не может работать.
- Он все запоминает зрительно и не стремится систематизировать.
- Зачем я буду ему платить, если могу всё сделать самостоятельно и забрать все деньги себе?

Основная ошибка начинающего руководителя в том, что он стремится сделать все сам, поскольку у него это получается быстрее, лучше, а также поскольку он несет ответственность.

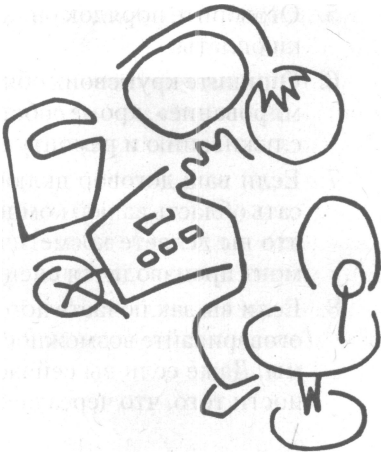
Для того чтобы свести риск провала работы со студентом к минимуму, следует делегировать студенту полномочия на черновую работу (ту, где только дурак может ошибиться), а себе оставлять заключительную сборочную часть.

Вот некоторые положительные моменты во взаимодействии со студентами:

- Студенты дешевы (а порой даже бесплатны).
- Студенты вполне могут согласиться работать за идею или полученный опыт.
- Студентам не нужно платить постоянно. Их можно почти мгновенно организовывать и почти мгновенно распускать.
- Студенты привыкли учиться, поэтому то, что вы будете говорить им, они будут с готовностью воспринимать.

Вы — волонтер!

Представьте ситуацию. Вьюокончили какое-либо учебное заведение и пошли работать. Как вы думаете, что у вас спросят первым делом? Какие специальности изучали и какие оценки получали? Как бы не так! Например, бухгалтера спросят о том, сколько и где он работал, имеет ли навык работы в «1С» и на компьютере. Если же бухгалтер никогда нигде не работал, как его могут принять на бухгалтерскую должность? Псевдобухгалтер будет работать оператором или кассиром. А если он пойдет работать бухгалтером, тогда что остается сказать о той профессии, для овладения которой не нужна подготовка? Точно так же программист не может быть программистом, если он не писал программ, а менеджер не может быть менеджером, если он не управлял.



Как же быть? Один из выходов — поиск тех, кто согласится «за так» обучать вас премудростям выбранной вами профессии. Когда вы найдете такого человека, то вы сэкономите мешки денег и вагоны времени. Чем больше вы будете работать бесплатно, тем большие преимущества вы приобретете.

В редакции одного журнала была сильная текучка кадров: только обучили очередного сотрудника — человек уходит, потому что зарплата маленькая. Так знаете, что придумали? Назвали работу обучением, зарплату — стипендией, а в конце выдавали диплом. И народ пошел. Вот вам цена обучения.

Приведу еще один пример из жизни. Одна немолодая бухгалтер однажды обмолвилась, что когда она наконец-то закончила вуз, ей стали платить больше. В этом примере всё на своих местах. Человек получил образование не ради образования, а для того, чтобы работодатель дороже оценивал его способности. Подумайте, согласился бы работодатель платить больше денег, если бы она получила высшее образование в области программирования?

Вы думаете, что я призываю бросать институты? Нет, нет и нет! Я призываю неустанно искать возможность приложения сил. Работайте бесплатно, но работайте!

Что следует помнить при заключении договора?

Необходимо грамотно организовать свою работу и правильно оформлять договоры. Кстати, договор необязательно должен быть письменным, он может быть и устным. Договор заключается не для того, чтобы судиться в случае невыполнения условий (да и вряд ли вы решитесь обратиться в суд), а для того чтобы исключить неоднозначные позиции относительно одних и тех же вещей. В конце концов, если клиент не захочет платить денег, то вы из него не вытрясете ни гроша даже при наличии договора.

1. Заключайте разные договора на разные этапы внедрения системы: договор на продажу коробки, договор на разработку костяка программы, договор на абонентское обслуживание, договор на обучение персонала.
2. Определите ответственного работника заказчика, который будет проверять сделанную вами работу и вести журнал учета отработанного вами времени.
3. Оговаривайте свои действия в случае смены части сотрудников (скажем, в случае их увольнения). Например, можно оговорить, что вы проводите обучение, но не больше четырех часов на одно рабочее место.
4. Завышайте сроки исполнения задач в два раза, В вопросе сроков исполнения договора лучше надевать очки пессимиста, чем оптимиста.

5. Оговорите порядок оплаты работы и порядок разрешения проблем задержки оплаты.
6. Опишите круг своих обязанностей. Возможно, клиент в понятие «программирование», кроме собственно программирования, включает работы по обслуживанию и ремонту компьютеров, обучение и поддержку веб-сайтов.
7. Если ваш договор включает обслуживание компьютеров, то следует описать область вашей компетенции, например, включив в договор пункт о том, что вы делаете косметический ремонт компьютеров, а более сложный ремонт производится специализированными фирмами.
8. Если вы заключаете договор на обслуживание программного обеспечения, оговаривайте возможность смены вас другим программистом из вашей фирмы. Даже если вы сейчас работаете в одиночку, это не исключает возможности того, что через некоторое время у вас появятся подчиненные.

Обучение как упражнение предпринимательства

Есть поговорка: обучал другого, пока сам не понял. В соответствии с этой поговоркой я, пока писал книгу, вновь разобрался в оборотных регистрах. Объясняя суть НДС, мне пришлось встретиться с несколькими бухгалтерами, чтобы они подтвердили правильность моих рассуждений. Обучая, вы волей-неволей систематизируете свои знания.

Обучение может быть одним из практических упражнений по приобретению навыков руководства. Старайтесь как можно раньше и больше обучать других: это нужно скорее вам, а не им. Обучая других, вы учитесь психологии управления.

Человек, который обучается у вас, психологически готов выполнять ваши приказания. То есть в вашей микрогруппе вы — начальник, а обучаемый — подчиненный. Кроме того, в процессе обучения вы ищете наиболее подходящие для убеждения слова, то есть учитесь убеждать.

Примеры предпринимательских решений

Ниже приведены предпринимательские решения, которые соответствуют принципу «Получи результат без денег».

Вопрос: Как участвовать в семинаре без оплаты?

Ответ: Предложить ведущему семинара услуги в области «вербовки» его участников.

Вопрос: Как участвовать в крутой конференции без оплаты?

Ответ: Быть не среди слушателей, а среди делающих доклады.

Вопрос: Как получить деньги от клиента, на обслуживание которого не хватает сил?

Ответ: «Продать» клиента фирме, которая выполняет схожие с вашими работы.

Вопрос: Как обучать новых сотрудников, не затрачивая при этом своих сил?

Ответ: Прикрепить ученика к какому-либо работнику и объяснить ему, что он предприниматель, который должен научиться руководить.

Вопрос: Как получить оценку на экзамене, не сдавая экзамен?

Ответ: Следует втереться в доверие к преподавателю, допустим, написать ему компьютерную программу.

Вопрос: Как найти новых клиентов, не затрачивая ни рубля?

Ответ: Попросить рекомендации у существующих клиентов.

Вопрос: Как бесплатно отработать методику обучения?

Ответ: Найти пару студентов и испытать методику на них.

Вопрос: Как снять помещения, не оплачивая их?

Ответ: Заключить бартер с хозяином помещения: он вам — угол, а вы ему — компьютерные услуги.

Вопрос: Как написать книгу с минимумом затрат?

Ответ: Предложить возможным авторам написать статьи, отзывы. Далее отзывы скомпилировать в книгу. Можно для этих целей использовать интернет-форумы и рассылки.

Подумайте — так, ради разминки — над следующими вопросами:

- Как я могу обучиться программированию без оплаты?
- Как я могу прямо сейчас превратить свои навыки в деньги?
- Как я могу сделать бизнес, не делая бизнес?
- Как я могу получать деньги, которые клиент платит безналичной формой, если у меня нет расчетного счета?

Как получить выгоду, не шевельнув пальцем?

Думаю, что получение положительного результата без вложения денег — это высший пилотаж предпринимательства. Приведу пример, который свидетельствует о том, что это возможно.

Студент попросил подработку на лето. К сожалению, в это время у меня не было фронта работ, который соответствовал бы его уровню. Можно было бы отказаться от идеи использовать студента, однако меня соблазнял тот факт, что он был сильным программистом в Visual C. Чтобы не упустить возможность, я решил выяснить, у кого из моих клиентов на лето уходит в отпуск системный администратор. На его место под мои гарантии был устроен студент.

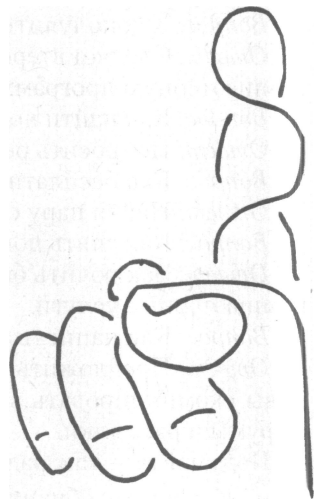
Со студентом я заключил соглашение, что за предоставление места работы и консультации в области администрирования сети и «1С» он мне напишет утилиту, которая подключалась бы к Internet Explorer.

Таким образом, благодаря одному предпринимательскому решению оказались довольными целых четыре стороны:

- Администратор смог уйти в отпуск и не искать себе замену.
- Работодатель был уверен, что его компьютерная сеть будет работать.
- Студент получил практику на один месяц и получил за нее деньги.
- Я получил необходимую мне программу.

Возможностей применить предпринимательский подход много, нужно лишь увидеть их.

Как за десять слов получить 800 рублей?



Приведу пример того, как компьютерщики могут получать доход за несколько правильно сказанных слов. Помните притчу об одном ударе молотком в нужном месте, за который специалист попросил 100 долларов? Этот подход можно распространить и на вас. Вы удивляетесь? Тогда проанализируйте факты.

Факт первый: каждая компьютерная фирма готова выплачивать агентский процент за информацию о том, кому можно продать компьютер.

Факт второй: если ваша работа связана с компьютерами (допустим, вы программист), то вас регулярно будут спрашивать о том, какой компьютер купить.

Попробуйте придумать, как вы можете синтезировать два эти факта и извлечь выгоду. Например, желающему купить компьютер можно сказать следующую фразу: «Если желаете, то *я помогу вам выписать счет*». За то, что вы правильно проанализировали ситуацию и сказали правильные слова, вы можете получить 4 % от продажи той техники, которую купит тот, кому вы рекомендовали свои услуги.

Расширьте рамки своего мышления за ваш профессиональный круг! Попытайтесь включить в схему анализа все факты, известные вам. Задайте вопрос, как я могу извлечь стократный эффект из создавшейся ситуации, но так, чтобы вложить минимум средств.

Успешные люди чаще других задают вопросы о том, как, вложив меньше средств, получить больший эффект.

Вот еще пример. На предприятии, состоявшем из нескольких торговых отделов, работал приходящий программист. Случилось так, что программист сиднем просиживал в одном отделе. Из этого же отдела шел сплошной поток нареканий в его адрес. После обсуждения с программистом этой конфликтной ситуации руководитель предприятия собрал работников злополучного отдела и задал несколько вопросов, что дало долгосрочный положительный эффект.

Вопросы были следующими:

- Как удается другим отделам, общаясь с программистом в течение нескольких часов в месяц, решать все проблемы?
- Как сделать так, чтобы работники отдела видели программиста в течение трех часов в месяц?
- Кто или что мешает программисту быстро выполнять свою работу?

Три вопроса, но каких!

Эффективные решения находятся рядом. Одним из способов найти их является формулировка «открытого вопроса», например: «Как справиться с задачей, приложив минимум усилий» или «Как решить проблему, используя личные связи».

Как обучать, не обучая?

Возможно, вы работаете один и вам предстоит сотрудничество с крупным заказчиком. У заказчика есть несколько отделов, где работают несколько десятков человек. Вам предстоит обучить их всех пользоваться новой программой. Возникает творческая проблема: как научить всех, если ваших сил явно не хватает.

Вы можете пригласить обучать базовым понятиям программы представителей учебных центров. Однако вы рискуете оказаться без клиента, поскольку обучающий центр может убедить заказчика в том, что вы несолидный исполнитель, и подменить вас своими программистами.

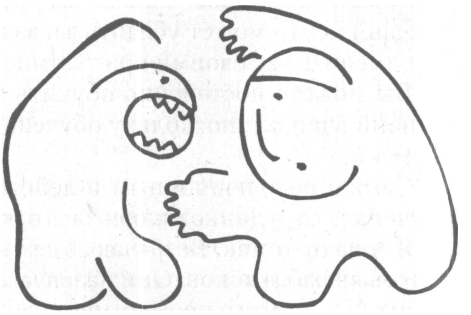
Вы можете постепенно обучить всех работников самостоятельно, но это не лучший вариант, поскольку обучение растянется на непредсказуемо длительное времени.

Самым революционным и действенным решением, на мой взгляд, является обучение сотрудников клиента его же сотрудниками.

Я делаю это так. Встречаюсь с руководством и выявляю нескольких наиболее толковых работников. Они назначаются ответственными за внедрение новой системы. После этого программист работает в основном только с этими ответственными работниками, а дальнейшую учебу они производят сами. Эффект от внедрения такой схемы обучения очевиден:

- Стоимость внедрения программы снижается, так как нет необходимости нанимать людей со стороны.
- Скорость внедрения изменений растет, поскольку программист концентрирует усилия на приоритетных направлениях.
- Коллектив становится более сплочелным, поскольку процветает система взаимопомощи.
- Программиста не дергают по пустякам, так как есть к кому обратиться внутри коллектива.

Глава 8
Удаленное обучение



Традиционная форма обучения основывается на лекционных занятиях, лабораторных работах, коллоквиумах и семинарах. Такое обучение ведется по программам, которые составлялась десятки лет назад. Утверждается, что такая форма обучения дает общие знания и основы системного мышления. Однако данная система обучения не решает проблему освоения насущных, пользующихся спросом навыков, для приобретения которых еще не составлен академический курс.

Возможны следующие варианты самообразования:

- самостоятельная работа с книгами,
- работа «подмастерьем» у специалиста,
- тусовки на форумах,
- обучение через Интернет (на семинарах) по методикам, разработанным практиками.

Казалось бы, тусовки (например, в студенческом кругу, среди специалистов какой-либо фирмы, на интернет-конференциях, форумах, семинарах) могут восполнить то, чего не найти в книгах. Однако обучающийся сталкивается с тем, что на его вопрос могут ответить, только если он четко сформулирован.

Есть еще одна проблема. Специалисты не тусуются в форумах, им не интересна публика, которая крутится там. Узнать что-либо новое или полезное для себя они не могут, поэтому обучившиеся специалисты (если только они не являются энтузиастами форумов или преподавателями курсов) постепенно вымываются из подобных образований. Другая проблема заключается в том, что оптимизацией работы новичка на форумах никто заниматься не будет, потому что мало кто из тусующихся может обладать видением проблемы.

Итак, требования к идеальной обучающей системе можно сформулировать следующим образом.

- С одной стороны, обучающая система должна давать академические знания, которые позволяли бы видеть общую картину, а не выборку ее фрагментов.
- С другой стороны, обучающая система должна обеспечивать прикладными знаниями, чтобы помочь обучающемуся решить любой практический вопрос.

Эти противоречия решаются использованием интернет-технологий. Все большее и большее количество специалистов открывают виртуальные курсы обуче-

ния. Обучение строится на принципе «мастер — ученик». Оттачивание мастерства идет в процессе решения конкретных задач, которые либо стоят перед учеником, либо предлагаются мастером. Мастер, как опытный лоцман, проводит ученика мимо множества подводных рифов и показывает, как правильно добиваться нужного результата. Мастер на проверенных опытом примерах объясняет азбуку системных знаний.

Конечно, ученик может и сам добиться нужного результата методом проб и ошибок. Однако беда новичка заключается в том, что он не всегда правильно оценивает задачу (сложное принимается за простое, а простое — за сложное), не может сформулировать вопрос (четко сформулированное условие — это две трети решения задачи), не знает, как проверить результаты своей работы (ведь он может изначально двигаться в ложном направлении и даже не подозревать об этом). Эти проблемы в одиночку решаются только временем. Нет ничего сверхмудрого в том, что освоили другие. Пройдет пять лет, и новичок будет недоумевать, почему он не понимал раньше таких элементарных вещей. Однако те, кому дорого время, выбирают обучение у специалистов.

Примеры обучающих заданий

Приведу примеры реальной переписки между преподавателем и обучающимся (студентом).

Работа со справочниками

Преподаватель: Требуется составить программу, которая перебирала бы справочник Товары и выводила на экран наименование товара и его ставку НДС. Посылаю вам шаблон кода, который следует проанализировать и переделать:

```
Процедура Сформировать()
спр = СоздатьОбъект("Справочник.Контрагенты");
спр.ВыбратьЭлементы();
Пока спр.ПолучитьЭлемент() = 1 Цикл
   Сообщить("" + Спр.Наименование + " " + Спр.Код);
КонецЦикла;
КонецПроцедуры
```

Студент:

```
Процедура Сформировать()
спр = СоздатьОбъект("Справочник.Товары");
спр.ВыбратьЭлементы();
Пока спр.ПолучитьЭлемент() = 1 Цикл
   Сообщить("" + Спр.Наименование + " " + Спр.Код);
КонецЦикла;
Сообщить(Константа.ПолучитьАтрибут("СтавкаНДС"));
КонецПроцедуры
```

Преподаватель: Программа написана почти правильно. Следовало заменить Контрагенты на Номенклатура. И еще: примеры следует проверять на работоспособность. В «Торговле» 9.** нет справочника Товары!

Студент:

```
Сообщить(Константа.ПолучитьАтрибут("СтавкаНДС"));
```

Преподаватель: Задание сделано некорректно. Я просил выдать каждое значение НДС товара (то есть много значений, а не одно значение). Кстати, ставка НДС прописывается в карточке товара. Посмотрите в Конфигураторе раздел Справочники.

Прошу изменить код.

Студент: Я не понял задания. Наверное, так? Но у меня не получилось.

```
Пока спр.ПолучитьЭлемент() = 1 Цикл
    Сообщить("" + Спр.Наименование + " " + значение НДС);
КонецЦикла;
```

Преподаватель: Правильно было бы написать так:

```
Пока спр.ПолучитьЭлемент() = 1 Цикл
    // Сообщить("" + Спр.Наименование + " " + значение НДС);
    Сообщить("" + Спр.Наименование + " " + Спр.СтавкаНДС);
КонецЦикла;
```

Для того чтобы понять, что следует писать СтавкаНДС, а не НДС, надо открыть структуру справочника и изучить, какие поля там есть. В числе других полей там есть поле СтавкаНДС.

Кстати, вот это:

```
Сообщить(Константа.ПолучитьАтрибут("СтавкаНДС"));
```

правильно писать так:

```
Сообщить(Константа.СтавкаНДС);
```

Реализация складского учета в продажных ценах

Преподаватель: Требуется организовать регистр, который учитывал бы, сколько товаров по определенной цене хранится на складе. За основу регистра можно взять регистр остатков.

Студент: Ничего не понял.

Преподаватель: Регистр остатков позволяет отвечать на следующие вопросы: сколько у данной фирмы — на данном складе — данного товара — хранится в штуках (я специально построил предложение так, чтобы отобразить структуру регистра).

При этом структура регистра будет следующая.

Измерения:

- Фирма,
- Склад,
- Товар.

Ресурсы:

- Количество.

Подумайте, куда следует внести еще одно поле Цена — в измерения или в ресурсы. Попробуйте проанализировать структуру других регистров.

Студент: Наверное, следует цену занести в ресурсы.

Преподаватель: Неправильно. Следовало построить предложение по предложенному мной шаблону, тогда все получилось бы как надо, то есть так: сколько у данной фирмы — на данном складе — данного товара — по данной цене — хранится в штуках.

Студент: А почему нельзя цены записать в ресурсы?

Преподаватель: Потому что ресурсы могут ответить на вопрос «сколько?». Например: «сколько цен?», «сколько штук?», «сколько рублей?» А вам нужно отвечать на вопрос «какие?» (какие товары, какие фирмы, какие цены). На вопрос «какие?» отвечают измерения регистра.

Работа с таблицей значений

Преподаватель: Теперь, опираясь на сконструированный в предыдущем примере регистр, напишите программу, которая сообщала бы результат для каждого товара в формате:

Цена, Количество.

Студент: Скажите, я двигаюсь в верном направлении?

```
Рег = СоздатьОбъект("Регистр.ОстаткиСЦенами");
остК = Рег.Остаток(Фирма,Склад,Товар,Цена,"Количество");
Сообщить("" + остК);
```

Преподаватель: Направление вы выбрали верное. Посмотрите команду ВыбратьИтоги. В вашей программе вы нашли одно решение, а требуется найти все цены.

Студент: Вот так можно найти все цены по выбранной фирме, по выбранному складу и по выбранному товару:

```
ПроцедураСформировать()
Рег = СоздатьОбъект("Регистр.ОстаткиСЦенами");
Рег.УстановитьФильтр(ВыбФирма,ВыбСклад,ВыбНоменклатура);
рег.ВыбратьИтоги();
Пока Рег.ПолучитьИтог() = 1 Цикл
    Сообщить("" + Рег.Товар + " " + Рег.Цена + " " +
Рег.Количество +
      " " + Рег.ТекущийДокумент());
КонецЦикла;
КонецПроцедуры*
```

Преподаватель: Замечательно! Теперь следует модифицировать программу так, чтобы она корректно списывала товар. Допустим, в регистре по конкретному товару хранится следующая информация:

Цена	Количество
10 р.	15 шт.
11 р.	70 шт.
12 р.	30 шт.
Итого:	115 шт.

Вашей программе требуется списать 100 штук товара так, чтобы товар списывался в следующей очередности:

Цена	Количество
Юр.	15 шт.
11 р.	70 шт.
12 р.	15 шт.
Итого:	100 шт.

Студент:

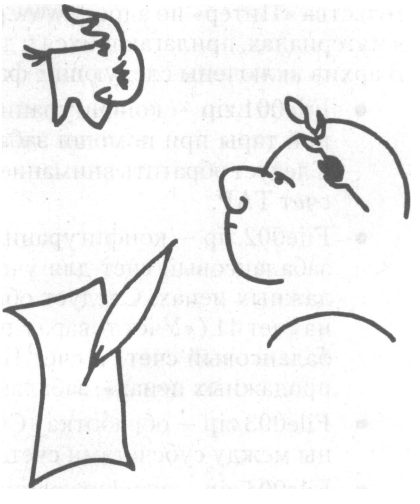
```
Процедура РазобратьПоЦенам()
// Создаем таблицу значений результатов обработки.
ТЗн = СоздатьОбъект("ТаблицаЗначений");
ТЗнРез = СоздатьОбъект("ТаблицаЗначений");
ТЗнРез.НоваяКолонка("НомерСтроки");
ТЗнРез.НоваяКолонка("Номенклатура");
ТЗнРез.НоваяКолонка("Количество");
ТЗнРез.НоваяКолонка("Единица");
ТЗнРез.НоваяКолонка("Коэффициент");
ТЗнРез.НоваяКолонка("Цена") ;
// Создаем таблицу значений остатков товаров.
ТЗн = СоздатьОбъект("ТаблицаЗначений") ;
// Выгружаем остатки регистра цен в таблицу значений.
Рег = СоздатьОбъект("Регистр.ОстаткиСЦенами");
ВыгрузитьТабличнуюЧасть(ТЗн) ;
КолСтр = ТЗн.КоличествоСтрок() ;
ТЗн.ВыбратьСтроки() ;

Пока ТЗн.ПолучитьСтроку() = 1 Цикл
    Рег.УстановитьФильтр(Фирма,Склад,ТЗн.Номенклатура);
    // Рассчитываем требующееся для списания количество товара.
    Кол = ТЗн.Количество*ТЗн.Коэффициент;
    рег.ВыбратьИтоги();
    Пока Рег.ПолучитьИтог() = 1 Цикл
        // Сообщить("" + Рег.Товар + " " + Рег.Цена + " " +
        // Рег.Количество;
        ЕстьКол = Рег.Количество;
        // Если остаток в регистре цен больше требующегося,
        // списываем остаток и прерываем цикл.
        Если ЕстьКол > ТЗн.Количество Тогда
            ТЗнРез.НоваяСтрока();
            ТЗнРез.Номенклатура = ТЗН.Номенклатура;
            ТЗнРез.Количество = ТЗН.Количество;
            ТЗнРез.Единица = ТЗН.Номенклатура.ОсновнаяЕдиница;
            ТЗнРез.Коэффициент = 1;
            ТЗнРез.Цена = Рег.Цена;
            Прервать;
        // Если остаток в регистре цен меньше требующегося,
        // списываем остаток из регистра
        //и уменьшаем требующееся для списания количество.
        Иначе
            ТЗнРез.НоваяСтрока();
            ТЗнРез.Номенклатура = ТЗН.Номенклатура;
```

```
                ТЗнРез.Количество = ЕстьКол;
                ТЗнРез.Единица = ТЗН.Номенклатура.ОсновнаяЕдиница;
                ТЗнРез.Коэффициент = 1;
                ТЗнРез.Цена = Рег.Цена;
                ТЗн.Количество = ТЗн.Количество - ЕстьКол;
            КонецЕсли;
        КонецЦикла;
    КонецЦикла;
КонецПроцедуры
```

Преподаватель: Молодец! Все правильно.

Условия обучения у автора книги



Автор книги ведет индивидуальные курсы обучения 1С-программированию через Интернет. Для обучения необходимо иметь доступ к Интернету и возможность посылать и принимать объемы данных больше 5 Мбайт. Кроме того, для успешного обучения студенту желательно иметь навык печати вслепую.

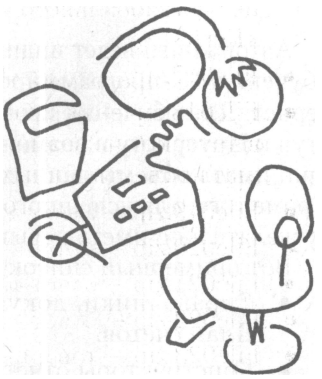
Вот примерный список тем обучения:

- Справочники, документы, регистры, План счетов.
- Конструкторы отчетов, запросов, бухгалтерских отчетов.
- Периодические величины, подчиненные справочники.
- Теория бухгалтерского учета для программиста.
- Обработка цепочки документов, например цепочки «расходная накладная счет-фактура — оплата».
- Запросы.
- Навыки постановки задачи программистом.

Конкретные условия обучения можно посмотреть на сайте www.prosto1c.by.ru.

Приложение А
Конфигурации, отчеты и обработки, размещенные в Интернете

Конфигурации, отчеты и обработки, упомянутые в тексте книги, вы можете найти на сайте издательства «Питер» по адресу www.piter.com/download в материалах, прилагающихся к данному изданию. В архив включены следующие файлы:



- File00l.zip — конфигурация учета возвратной тары при помощи забалансового счета. Следует обратить внимание на забалансовый счет ТАР.
- File002.zip — конфигурация, использующая забалансовый счет для учета товара в продажных ценах. Следует обратить внимание на счет 41 («Учет товара в покупных ценах», балансовый счет) и счет П («Учет товара в продажных ценах», забалансовый счет).
- File003.zip — обработка «Сбор мусора 62», решающая проблему пересортицы между субсчетами счета 62.
- File004.zip — конфигурация, превращающая товары (счет 41) в материалы (счет 10), преобразуя при этом количества. Следует обратить внимание на документ «Требование-накладная».
- File005.zip — конфигурация, реализующая работу со счетом 20.
- File006.zip — конфигурация, в которой реализуется розничный учет в суммовом выражении. Следует обратить внимание на документ «Товарные отчеты розницы».
- File007.zip — отчет, формирующий книгу продаж по оплате без использования стандартных регистров.
- File008.zip — обработка, анализирующая продажи с оптовой фирмы и формирующая расходную и приходную накладные.
- File009.zip — отчет по определению последних неоплаченных накладных.
- File010.zip — отчет «Таблица оплат расходных накладных в хронологическом порядке без использования регистров».
- FileOll.zip — конфигурация, в которой реализован многофирменный учет при помощи компоненты «Бухгалтерия».
- FileO12.zip — конфигурация, демонстрирующая загрузку файла с накладной в программу.

- FileO13.zip — конфигурация «Учет работы группы программистов».
- FileO14.zip — конфигурация, в которой запрограммирована схема из главы «Постановка задачи для предприятия X». Следует смотреть модуль проведения расходной и приходной накладных, а также приходного кассового ордера и банковской выписки.
- FileO15.zip — конфигурация, в которой реализованы товарные отчеты розничной торговли «К». В конфигурации следует смотреть документ «Розница».
- FileO16.zip — конфигурация, в которой реализован учет кег. Следует обратить внимание на следующие объекты конфигурации: регистр «Кеги», модуль документов «Расходная накладная» и «Приходная накладная», процедура глобального модуля «Кеги», документ «ИнвентаризацияКег», отчет «Кеги».
- FileO17.zip — отчет «Пакетная печать документов».
- FileO18.zip — обработка, переносящая документы из одной базы данных в другую (справочники Номенклатура и Контрагенты, расходные и приходные накладные, приходные кассовые ордера и банковские выписки).
- FileO19.zip — конфигурация «Помощник писателя».
- File020.zip — отчет «Товар - Клиент».
- FileO21.zip — отчет «Акт сверки».
- FileO22.zip — отчет «Лист загрузки в автомобиль».
- FileO23.zip — товарный отчет в «Бухгалтерии», но по форме «Торговли».

Приложение Б
Соглашение об условиях и порядке работы программиста

г. Хабаровск «___»_____200___г.

Настоящим соглашением определяются условия **и** порядок взаимоотношений между_____(далее «Заказчик») в лице _____**и**_____(далее «Специалист»).

1. Специалист выполняет работу по внедрению, **модернизации,** оптимизации и текущему функционированию программы «1С» в компьютерной сети Заказчика.
2. Специалист самостоятельно по согласованию с представителями Заказчика предлагает, а также по отдельным заявкам представителей Заказчика устанавливает, внедряет, отлаживает и налаживает наиболее рациональные и оптимальные варианты конфигурации «1С».
3. Специалист самостоятельно или по заявкам представителей Заказчика выполняет работу (но не более 30 % от общих объемов трудозатрат) по на ладке, настройке и мелкому ремонту компьютерной техники в рамках профессиональных знаний, соответствующих его уровню.
4. Специалист выполняет подбор третьих лиц и согласование с ними процедур и порядка действий при решении вопросов но функционированию компьютерной техники, находящихся вне его компетенции.
5. Специалист выполняет свою работу в объеме 288 часов в год, при этом отработка ежедневно фиксированного количества рабочего времени не является обязательной. Процесс работы Специалиста должен быть организован таким образом, чтобы не приводить к простоям в деятельности Заказчика.
6. Специалист обязан прибывать в офис Заказчика по телефонным заявкам в течение рабочего дня в согласованное время.
7. Заказчик обязан вести тетрадь учета посещений Специалиста с указанием дат, продолжительности работы **и** ее результатов под росписи заказчиков и Специалиста.
8. Заказчик обязан в согласованное время предоставлять Специалисту рабочее место.
9. Заказчик обязан оплачивать работу Специалиста из расчета_____рублей в месяц на условиях ежемесячной оплаты.
10. При необходимости Заказчик предоставляет Специалисту пропуск для въезда на территорию своей базы.
11. Заказчик может делать___экстренных вызовов Специалиста в месяц. Каждый вызов сверх этого количества оплачивается в сумме_____рублей за

вызов. Экстренным считается вызов, исполнение которого требует приезда Специалиста в течение менее чем 4 часов после получения заявки.

12. Заказчик предоставляет Специалисту доступ к компьютерной технике после 17.00, а до 17.00 — по согласованию с Заказчиком.
13. В случае расторжения договора Заказчик должен оплатить Специалисту фактически отработанное время из расчета среднечасовой оплаты.
14. Специалист может использовать для исполнения работ по договору третьих лиц.
15. Сумма по договору остается неизменной в случае, если количество отработанных Специалистом часов будет находиться в пределах 240-340 часов в год. При увеличении (уменьшении) количества отработанного времени свыше 340 (меньше 240) сумма договора увеличивается (уменьшается) из расчета среднечасовой оплаты.
16. Договор вступает в силу с момента его подписания и действует до «__» _____200_г.

Заказчик_____Специалист_____

Приложение В
Договор на обслуживание компьютерной техники и бухгалтерского программного обеспечения

Договор № _

г. Хабаровск «__»_____200_ г.

Предприятие_____(в дальнейшем «Исполнитель») в лице_____, действующего на основании _____, и_____(в дальнейшем «Заказчик»), в лице _____, действующего на основании _____, заключили договор о том, что Исполнитель берет на себя обязательства, а Заказчик поручает выполнение работ по обслуживанию бухгалтерского программного обеспечения.

1. Порядок выполнения работ.

Исполнитель должен посещать Заказчика не реже двух раз в неделю. Заказчик имеет возможность сделать не более двух срочных вызовов в месяц. Под срочными понимаются такие вызовы, исполнение которых должно происходить не позже, чем через два часа. Заявки производятся на мобильный телефон программиста, работающего от лица Исполнителя_____, или на телефон

2. Виды выполняемых **работ.**

- Обучение персонала бухгалтерской программе «1С».
- Поиск ошибок в базе данных и выработка мер но их устранению.
- Реконфигурация бухгалтерского программного обеспечения.

3. Оплата.

Оплата производится в размере_____рублей в месяц. Оплата производится ежемесячно в течение первой недели месяца.

4. **Сроки выполнения заявки.**

Договор действует в течение одного года, начиная с «__»_____200_ г.

В случае прекращения договора одна из сторон должна предупредить противоположную сторону о намерении прервать договор.

5. Реквизиты.

Исполнитель:_____

Заказчик:_____

Исполнитель Заказчик

Приложение Г
Пример проспекта программиста 1С-услуг

1С: Программирование
бухгалтерских систем

Программист

Сергей Михайлов

**Я знаю,
как сделать лучше!**

Вручную или на компьютере?

Внедрение сети торговых агентов, развертывание структуры филиалов, оптимизация налоговых выплат, минимизация запасов товара на складах, контроль дебиторской задолженности клиентов, проведение маркетинговых исследований, ведение еженедельного баланса предприятия — **всё это невозможно без применения компьютеров.**

Пара слов о себе

Я связан с оптовым бизнесом более семи лет. Сам сдавал бухгалтерские балансы. Работал финансовым директором. Работал программистом у дистрибьютора продукции Procter&Gamble на Дальнем Востоке. Участвовал в импортных поставках продуктов питания. Принимал участие в запуске нескольких агентских и филиальных структур. Внедрял систему ежедневных балансов предприятия. Участвовал в запуске системы консигнационных складов. Настраивал компьютерный бухгалтерский учет производственных фирм, оною особенности ведения акцизного товара.

Мне понятен язык, на котором говорят бухгалтер и начальник торгового отдела!

Сергей Михайлов
тел.....

иергей Михайлов
тел.....

Порядок работы с клиентом

Связь с клиентами ведется с помощью пейджера. Вы звоните по телефону _____и кратко излагаете свою проблему, например так:

> Сергей, не могу рассчитать задолженность клиентов. Лена из «АРСУ», тел.... не срочно.

Или так:

> Все пропало! Выключили свет. Тел. ... Сергей, подъезжайте СРОЧНО!

Или совсем просто:

> Пожалуйста, позвони по тел:.., Лариса.

Пусть будет бизнес более управляемым, а учет — менее сложным!

Сергей Михайлов
тел.....

Экономический эффект от внедрения компьютерного учета

- Применение компьютерного учета в агентской сети увеличивает информированность руководства о дебиторской задолжности, что позволяет сократить сроки оплаты клиентов с 13 дней до 10 дней, что приводит к **сокращению задолженности на 30** %.
- Применение компьютерного учета в агентской структуре позволяет внедрить систему соревнований между агентами на лучшие продажи товара, что повышает объемы продаж неликвидного товара на 25 %. После внедрения компьютерной системы **вы сможете избавляться от залежавшегося товара не снижением цены, а внедрением системы соревнований между агентами.**
- Внедрение компьютерного учета позволяет вести оперативный учет товара на складах и иметь мгновенный срез остатков товара на конкретную дату. Это позволяет делать экономически обоснованный заказ товара. Внедрение практики компьютерного планирования остатков товара позволяет **сократить товарный остаток на 5-18%.**

Сергей Михайлов
тел.....

Формуляр для определения варианта сотрудничества:

- Наличие 1С:Торговля
- Наличие 1С:Бухгалтерия
- Количество компьютеров
- Количество компьютеров, работающих под 1С
- Как часто вы хотели бы, чтобы программист посещал ваше предприятие?

Контактная информация:

Моб. тел. ...
E-mail: mik(a)mail.kht.ru
http:// www.prof.narod.ru
Тел.:

Основная специализация

- Почасовая работа специалиста по настройке программ 1С:Торговля и склад
- Договор на месячное обслуживание программы 1С
- Внедрение бухгалтерского и торгового компьютерного учета с нуля

Дополнительные виды работ

- Прокладка локальных сетей
- Настройка работы терминальной сети
- Обучение работе в 1С
- Обслуживание компьютерной техники
- Интернет-проекты

Сергей Михайлов
тел.....

Сергей Михайлов
тел.....