Наталья Рязанцева, Дмитрий Рязанцев

bhv®

1С:Предприятие

Секреты программирования

- Конфигуратор и структура информационной базы
- Объектно-ориентированное программирование в 1С
- Атрибуты и методы метаданных
- Обработка внешних событий
- Восстановление базы данных

МАСТЕР

Наталья Рязанцева

Дмитрий Рязанцев

1С:Предприятие

Секреты программирования

Санкт-Петербург

«БХВ-Петербург»

2004

УДК 681.3.06
ББК 32.973.26-018.2
Р99

Рязанцева Н., Рязанцев Д.

Р99 1С:Предприятие. Секреты программирования. — СПб.: БХВ-Петербург, 2004. — 352 с.: ил.

ISBN 978-5-94157-416-2

Книга посвящена изучению приемов и методов программирования на встроенном языке "1С:Предприятие" с применением компонент, используемых при разработке конфигураций "Бухгалтерский учет", "Оперативный учет" и "Расчет". Для лучшего понимания логики программирования приведена структура информационной базы всех трех компонент. Наряду с начальными сведениями о базовых конструкциях языка и встроенных средствах проектирования пользовательского интерфейса, обсуждаются методы восстановления базы данных и средства обработки внешних событий. Рассмотрены стандартные приемы программирования, используемые при модифицировании уже существующих конфигураций. Представлены схемы построений модулей, полный объем атрибутов и методов метаданных с уникальными примерами разработки конфигурации "Коммунальные услуги". Описан механизм обработки внешних событий на примере сканера штрих-кода. Рассматриваются ошибки и трудности, которые могут возникнуть в процессе отладки и работы программ.

Для 1С-программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. гл. редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Владимир Красильников*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 09.06.04.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 28,4.
Тираж 5000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

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

ISBN 978-5-94157-416-2

Содержание

Глава 1. Общий алгоритм работы программного обеспечения 1

Основные файлы конфигурации 1
Структура файла 1Cv7.md 1
Структура файла 1Cv7.dd 10
Основные файлы базы данных 15
Общие обязательные таблицы для всех конфигураций 16
Файлы компоненты "Расчет" 22
Файлы компоненты "Оперативный учет" 24
Структура файла 1sstream 27
Файлы компоненты "Бухгалтерский учет" 28
Структура файла 1saccs.dbf 28
Структура файла 1soper.dbf 31
Структура файла 1sentry.dbf 32
Структура файла 1sbkttlc.dbf 35
Структура файла 1sbkttl.dbf 37
Структура файла 1scorent.dbf 39
Структура файла 1saccsel.dbf 40
Структура файла 1stoper.dbf 42
Структура файла 1ssbsel.dbf 43
Файлы компоненты "Расчет" 44

Глава 2. Программирование на встроенном языке "1С:Предприятие" 49

Введение в объектно-ориентированное программирование 49
Программирование на языке 1С 50
Синтаксис и конструкции встроенного языка 52
Процедура 53
Функция 56
Специальные символы 58
Оператор условного выполнения 59
Оператор цикла 62
Оператор *ПопыткаИсключение* 62

Оператор *ВызватьИсключение* 65
Оператор *Перейти* 66
Оператор *Продолжить* 67
Оператор *Прервать* 68
Оператор *Возврат* 68
Директивы 69
Алгоритм исполнения модулей встроенного языка 71
Процедуры и функции элементов формы 71
Предопределенные процедуры 74
Глобальный модуль 109
Навигация в теле модуля 121
Синтакс-Помощник 122

Глава 3. Атрибуты и методы метаданных 125

Атрибуты метаданных 125
Атрибуты справочников 125
Атрибуты документов 127
Атрибуты операций и проводок 130
Атрибуты регистров 133
Атрибуты видов расчетов 136
Атрибуты правил перерасчета 137
Атрибуты календаря 138
Атрибуты счетов 139
Атрибуты журнала расчетов 141
Общие методы объектов метаданных 142
Методы с использованием функции *СоздатьОбъект()* 153
Функция *СоздатьОбъект()* 154
Методы документов 162
Методы справочников 168
Методы регистров 177
Методы операций 186
Методы объектов типа "БухгалтерскиеИтоги" 196
Методы таблиц значений 230
Методы таблиц 242
Методы журнала расчетов 258
Методы констант 262
Методы периодических реквизитов 262

Глава 4. Интерфейс с другими программными продуктами 263

Методы файловой системы 266
Методы объекта типа "XBase" 270
Методы работы с текстовыми файлами 278

Методы объекта типа "Текст" ... 278
Методы чтения и записи текстовых файлов ... 284

Глава 5. Работа над ошибками ... 287

Синтаксический контроль ... 287
Нарушение синтаксических конструкций ... 287
Отсутствие инициализации переменной ... 290
Синтаксический контроль запросов ... 291
Ошибки выполнения ... 291
Ошибки задания методов ... 291
Ошибки передачи параметров ... 292
Ошибки специфики объекта ... 293
Ошибки формирования таблиц ... 294
Синтакс-Помощник ... 294

Глава 6. Обработка событий ... 297

Формирование сообщений ... 297
Обработка внешних событий ... 300
Настройка оборудования ... 301
Обработка обслуживания ... 304
Обмен данными ... 310

Глава 7. Рекомендации по сопровождению программы "1С:Предприятие" ... 315

Восстановление базы данных ... 315
Некорректность бухгалтерских итогов ... 315
Тестирование и исправление информационной базы ... 317
Ошибка запуска Конфигуратора ... 319
Ошибка открытия файлов ... 319
Обновление конфигурации ... 320
Методические рекомендации по изменению конфигурации ... 323

Предметный указатель ... 329

Глава 1

Общий алгоритм работы программного обеспечения

Программисты часто сетуют на то, что структура информационной базы данных "1С:Предприятие", в традиционном понимании файловой системы, скрыта от них, что она сложна и непонятна. Ниже раскрывается схема работы программы и структура баз данных, для того чтобы программисты могли представить, как происходит в программе обработка информации.

Программное обеспечение "1С:Предприятие" организовано по принципу максимального разделения исполняющей системы и данных. Исполняющая система интерпретирует конфигурацию, которая задает интерфейс пользователя с базой данных (см. рис. 1.1).

Основные файлы конфигурации

Исполняющий модуль 1Cv7.exe использует для интерпретации два основных файла, описывающих конфигурацию системы: файл конфигурации (1Cv7.md) и словарь базы данных (1Cv7.dd). Исполняющий модуль следит за целостностью информации. Любые изменения конфигурации фиксируются в обоих файлах. Все программные процедуры конфигурации хранятся в файле 1Cv7.md в специальном формате и доступны программисту только в *Конфигураторе*. Разделение описательной информации на два файла довольно удобно. Оно освобождает программистов от формирования структуры базы данных. Взаимосвязь данных файлов видна в их структурах.

Структура файла 1Cv7.md

Информацию о конфигурации в файле можно рассматривать в двух аспектах — описание метаданных и обработки событий. Метаданные описываются отдельными фреймами, в которых вложены описания формы, реквизитов, элементов интерфейса и печатных форм, выделенные разделителями — {, }.

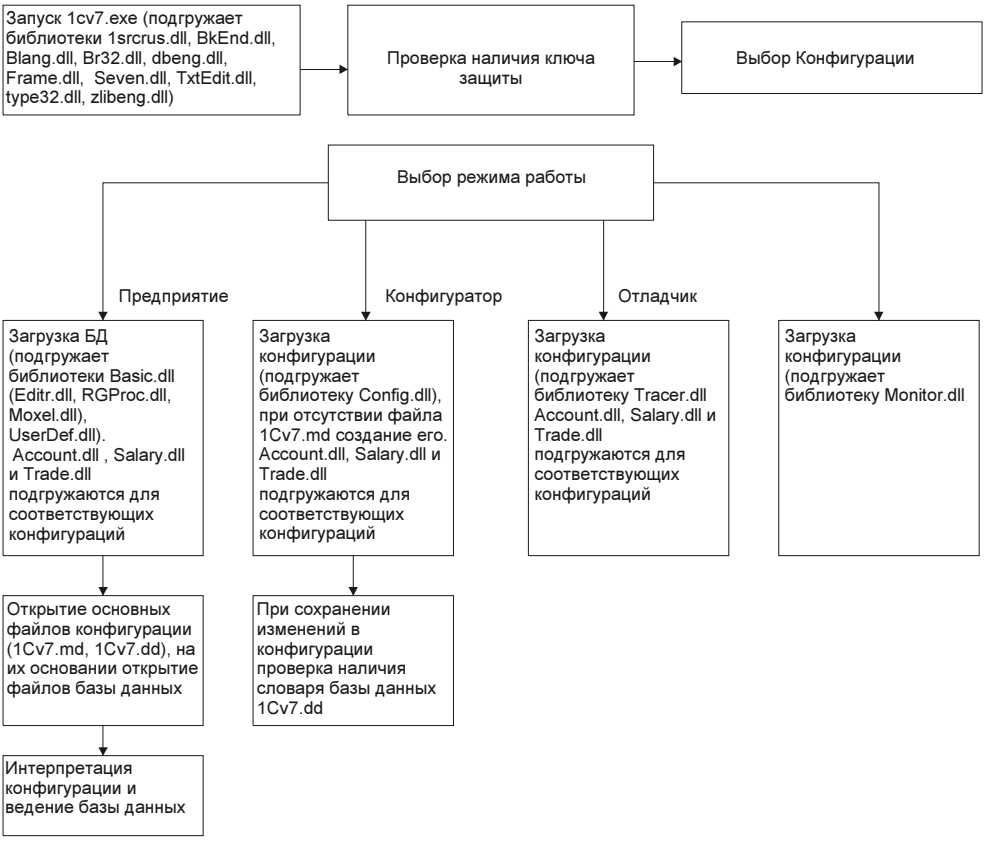


Рис. 1.1. Схема работы программы

Описание справочников

Рассмотрим структуру справочника "Валюты" (рис.1.2). В справочнике использованы следующие обозначения:

- Код — строка длиной 3 символа;
- Наименование — строка длиной 25 символов;
- ПолноеНаименование — строка длиной 50 символов;
- Курс — число длиной 10 с точностью 4 знака;
- Кратность — число длиной 7 знаков;
- ИмяФайлаПрописи — строка длиной 12 символов.

Описание справочника будет представлено следующим фрагментом:

{"Frame",

{"-11","0","0","0","400","0","0","0","204","1","2","1","34","MS Sans Serif","290","155",

"Валюты","","","0","","1","1","6","25","-1","0","0",

{"0",

{"Основной","1"}},"1","1"}},

{"Browser","0","1",

{"Multicolumn",

{"","browse","1353711616","8","19","275","99","0","0","4151","","","","0","US

{"Fixed",

{"3"," ","26","STATIC","4152","","","Пиктограмма","-
2567","0","0","0","0","0","2","","0","0","","","","0"},

{"2","Код","48","1CEDIT","4153","","","Код","-
2568","2","3","0","0","0","2","","0","0","","","","0"},

{"2","Наименование","94","1CEDIT","4154","","","Наименование","-
2569","2","25","0","0","0","2","","0","0","","","","0"},

{"1","Курс","71","1CEDIT","4155","","","Курс","99","1","10","4","0","1","2","","0",
"16","","Курс валюты на выбранную дату","","0"},

{"1","Кратность","76","1CEDIT","4156","","","Кратность","100","1","7","0","0","1",
"2","","0","16","","Кратность","","0"}}},

{"Controls",

{"...","BUTTON","1342177291","272","137","11","13","0","0","4152","","ПоКнопке
ВыбораДаты()","","-1","U","0","0","0","0","4","","По нажатию этой кнопки
будет открыт диалог

выбора даты просмотра значений периодических

реквизитов.","Ввести дату","0","-11","0","0","0","0","0","0","0","0","0","0","0","0","0",
"MS Sans Serif","-1","-1","0","Основной","{""0"","""0""}"},

{"Значения периодических реквизитов указаны на 01.01.1980","STATIC",
"1342177280","140","135","129","17","0","0","4153","","глДатаПериодРеквизитов
(ИспользоватьДату())","","-1","U","0","0","0","0","65600","","Дата, на которую
просматриваются

значения периодических реквизитов.","","0","-11",
"0","0","0","400","0","0","0","204","1","2","1","34","MS Sans Serif","8388608",
"-1","0","Основной","{""0"","""0""}"},

{"История","BUTTON","1342177291","81","137","54","13","0","0","4154","",
"ПоКнопкеИстория()","","-1","U","0","0","0","0","2064","","История изменения
периодических реквизитов","","0","-11","0","0","0","0","0","0","0","0","0","0","0","0",
"MS Sans Serif","-1","-1","268435494","Основной","{""7"","""116""}"},

{"","BUTTON","1342177291","64","137","15","13","0","0","4155","","Выбрать
ПравовуюСправку()","Кн_Справка","-1","U","0","0","0","0","1040","","Правовая

справка","","0","-11","0","0","0","0","0","0","0","0","0","0","0","0","0","MS Sans Serif","-1","-1","268435488","Основной","{""0"",""0""}"},

{"&Закрыть","BUTTON","1342177291","8","137","54","13","0","0","4156","","#Закрыть","","-1","U","0","0","0","0","0","","","Закрыть","0","-11","0","0","0","0","0","0","0","0","0","0","0","0","0","MS Sans Serif","-1","-1","0","Основной","{""0"",""0""}"},

{"Отчет по курсу валюты","BUTTON","1342177291","193","121","90","13","0","0","4157","","ПостроитьОтчет()","","-1","U","0","0","0","0","0","16","","История курсов выбранной валюты","","0","-11","0","0","0","0","0","0","0","0","0","0","0","0","0","MS Sans Serif","-1","-1","0","Основной","{""0"",""0""}"},

{"Добавить...","BUTTON","1342177291","137","121","54","13","0","0","4158","","ВыборИзКлассификатора()","","0","U","0","0","0","0","0","16","","Добавить из Классификатора валют","","0","-11","0","0","0","0","0","0","0","0","0","0","0","0","0","MS Sans Serif","-1","-1","0","Основной","{""0"",""0""}"},

{"Получить курсы валют с www.rbc.ru","BUTTON","1342177291","8","121","127","13","0","0","4159","","ОткрытьФорму(""Отчет.ИППКурсыВалютРБК"")","","-1","U","0","0","0","0","0","16","","Получить курсы валют с сервера www.rbc.ru","","0","-11","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","MS Sans Serif","-1","-1","0","Основной","{""0"",""0""}"}}

В форме задания параметров справочника, как показано на рис. 1.2, присутствуют два реквизита, которых нет в представленном выше фрагменте.

Реквизиты "ПолноеНаименование" и "ИмяФайлаПрописи" не присутствуют в интерфейсе основной формы справочника (см. рис. 1.3).

Отсутствующие реквизиты представлены во фрагменте, который приведен ниже.

{"104","Валюты","","","0","3","1","1","1","25","1","2","1","103","102","0","1","1",

{"Params",

{"98","ПолнНаименование","","Полное наименование","S","50","0","0","0","0","0","1","0","0","1","0","0"},

{"99","Курс","Текущий курс","","N","10","4","0","1","0","1","1","0","0","1","0","0"},

{"100","Кратность","Кратность","","N","7","0","0","1","0","1","1","0","0","1","0","0"},

{"101","ИмяФайлаПрописи","Имя файла прописи","","S","12","0","0","0","0","0","1","0","0","1","0","0"}},

{"Form",

{"102","Основная","Основная",""},

{"103","ДляВыбора","ДляВыбора",""}}}

Каждому объекту метаданных присваивается уникальный номер, по которому осуществляются все ссылки. В первом фрагменте видно, что номер справочника — 104, номер реквизита "Курс" — 99. При добавлении нового объекта

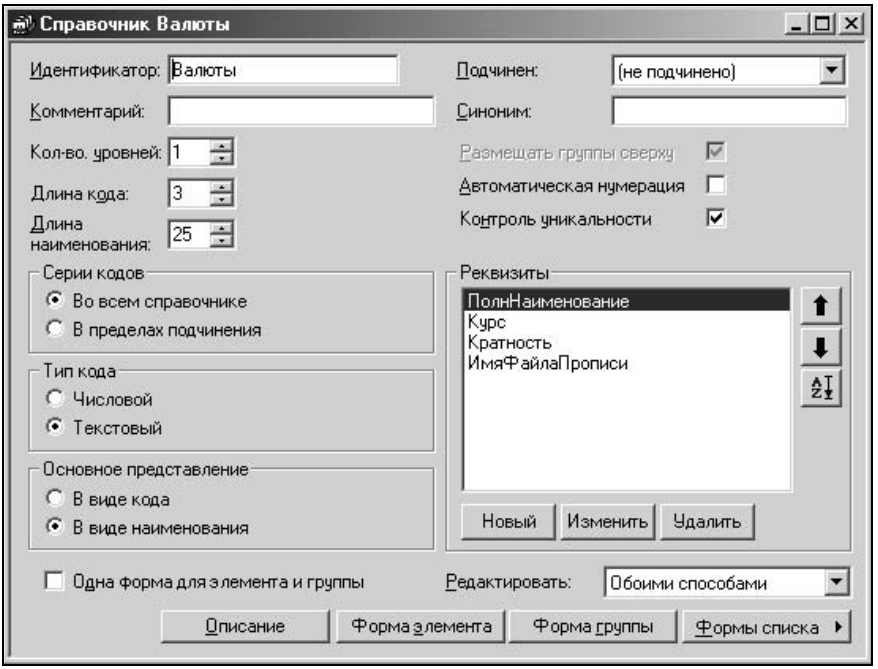


Рис. 1.2. Диалоговое окно настройки справочника "Валюты"

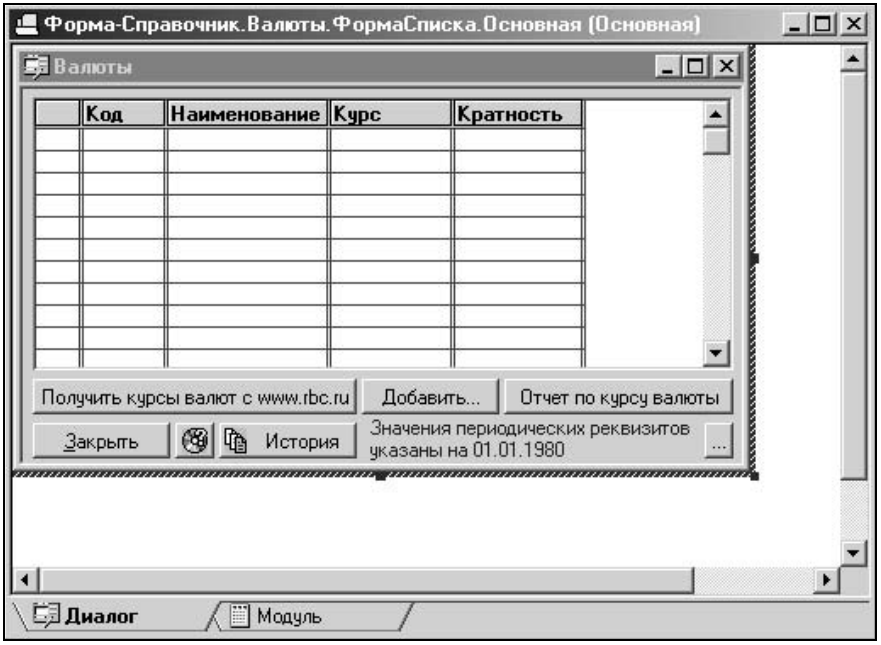


Рис. 1.3. Диалоговое окно настройки основной формы-справочника "Валюты"

или реквизита используется текущий номер, поэтому номера реквизитов в рамках одного объекта могут быть непоследовательными. Данные номера используются и в файле 1Cv7.dd.

Рассмотрим соответствие параметров, заданных в Конфигураторе, параметрам в приведенном фрагменте на примере реквизита "Курс" (см. рис. 1.4, рис. 1.5).

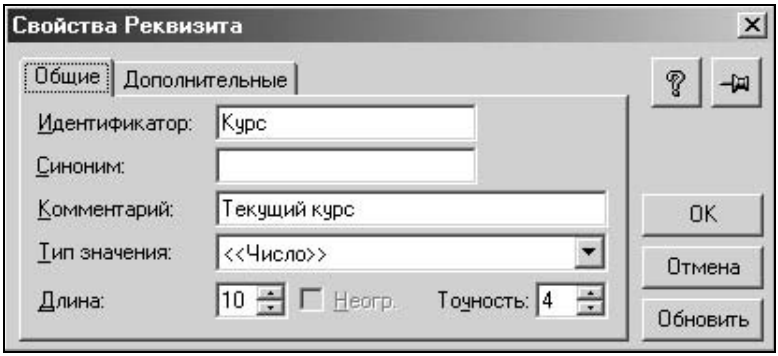


Рис. 1.4. Диалоговое окно **Свойства Реквизита** на вкладке **Общие**

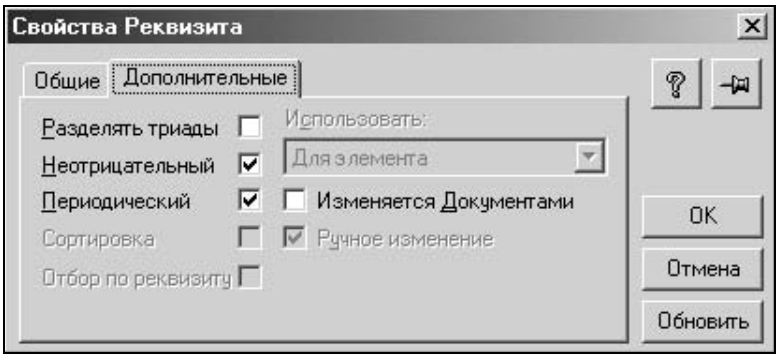


Рис. 1.5. Диалоговое окно **Свойства Реквизита** на вкладке **Дополнительные**

Из фрагмента видно, что:

- Номер объекта — первый параметр;
- **Идентификатор** — второй параметр после номера;
- **Комментарий** — третий параметр;
- **Синоним** — четвертый параметр;
- **Тип значения** — пятый параметр;
- **Длина** — шестой параметр;

- **Точность** — седьмой параметр;
- **Вид заданного типа реквизита** — восьмой параметр.

Тип значения в модуле, для данного случая, обозначен как — "N". Данный параметр может принимать различные значения. Это зависит от их типов, которые, в свою очередь, могут быть:

- S — строка;
- N — число;
- O — документ;
- B — справочник;
- E — перечисление;
- T — счет;
- U — неопределенный.

Конкретный вид заданного типа реквизита задается восьмым параметром. В данном случае используется все тот же номер объекта метаданных, присвоенный при создании объекта и хранящийся в описании объекта. Например, в справочнике "Виды номенклатуры" реквизит — "Тип номенклатуры" описан как объект типа "E" вида 452, т. е. с номером 452. Реквизит "Основной материал" описан как объект типа "B" вида 11307, т. е. справочник под номером 11307.

{"10951","ВидыНоменклатуры","","Виды продукции (работ, услуг)","0","7","1","1","2","60","1","0","3","10952","10952","0","1","1",

{"Params",

{"10954","ТипНоменклатуры","","Тип номенклатуры","E","0","0","452","0","0","0","1","0","1","1","0","1"},

{"27275","ОсновнойМатериал","Основное сырье в определении ст. 319 НК РФ","","B","0","0","11307","0","0","0","1","0","0","1","0","0"}},

{"Form",

{"10952","ФормаСписка","",""}}},

Остальные параметры (см. рис. 1.5) указываются на вкладке **Дополнительные** и нумеруются следующим образом:

- **Разделять триады** — девятый параметр;
- **Неотрицательный** — десятый параметр;
- **Периодический** — одиннадцатый параметр;
- **Для элемента** — двенадцатый параметр;
- **Не используется** — тринадцатый параметр;
- **Сортировка** — четырнадцатый параметр;
- **Ручное изменение** — пятнадцатый параметр;

- **Изменяется Документами** — шестнадцатый параметр;
- **Отбор по реквизиту** — семнадцатый параметр.

Описание документов

В описании структуры документа фрагмент, описывающий поля, имеет две составляющих:

1. *Шапка* — Head Fields.
2. *Табличная часть* — Table Fields.

Описание многострочной части документа не случайно выделено в отдельный сегмент. Табличная часть документа хранится в отдельном файле. Шапка документа также хранится в файле, но в другом.

Рассмотрим фрагмент с описанием документа "Расходная накладная".

{"294","РасходнаяНакладная","Отгрузка товаров","Отгр.товаров","6","1","1","2","500",
"-1","1","0","0","0","1",
{"38566","","","","",
{"Refers",
{"238"},
{"308"},
{"11012"},
{"16172"},
{"16167"},
{"12517"}}},"0","0","1","1","0","1",
{"Head Fields",
{"277","Контрагент","Контрагент","","B","0","0","133","0","0"},
{"278","Договор","Договор","","B","0","0","112","0","0"},
{"279","МестоХранения","МестоХранения","","B","0","0","135","0","0"},
{"280","ВидОтгрузки","1- на счет 90, 2- на счет 45, 3 - возврат поставщику (на счет76.2)","Вид отгрузки","N","1","0","0","0","0"},
"281","УчитыватьНП","не используется","","N","1","0","0","0","0"},
{"282","ЗачитыватьАванс","0- зачитывать аванс при проведении. 1- не зачитывать аванс","Зачитывать аванс","N","1","0","0","0","0"},
{"16096","Курс","","","N","10","4","0","1","0"},
{"16097","ВариантРасчетаНалогов","","","B","0","0","11147","0","0"},
{"16098","РасчетныйСчет","Расчетный счет","","B","0","0","97","0","0"},
{"16099","ВерсияОбъекта","Служебный реквизит","Версия объекта","S","8","0",
"0","0","0"},
{"27347","НДСвключатьВСтоимость","","","N","1","0","0","0","0"},

{"27348","ДокументПоступления","","","O","0","0","0","11188","0","0"}},
{"Table Fields",
{"283","Товар","Товар","","B","0","0","156","0","0","0"},
{"284","Количество","Количество","","N","14","3","0","0","0","0"},
{"285","Цена","","","N","15","2","0","0","1","0"},
{"286","Сумма","","","N","15","2","0","0","1","0"},
{"287","НДС","Сумма НДС","","N","15","2","0","0","1","1"},
{"289","НП","","","N","15","2","0","0","1","0"},
{"290","Всего","","","N","15","2","0","0","1","1"},
{"291","Комитент","","","B","0","0","133","0","0","0"},
{"292","ДоговорКомиссии","","Договор комиссии","B","0","0","112","0","0","0"},
{"27350","ГТД","","Грузовая таможенная декларация","B","0","0","223","0","0","0"}}}

В форме задания документа присутствуют две части параметров: **Реквизиты шапки** и **Реквизиты табличной части** (см. рис. 1.6).

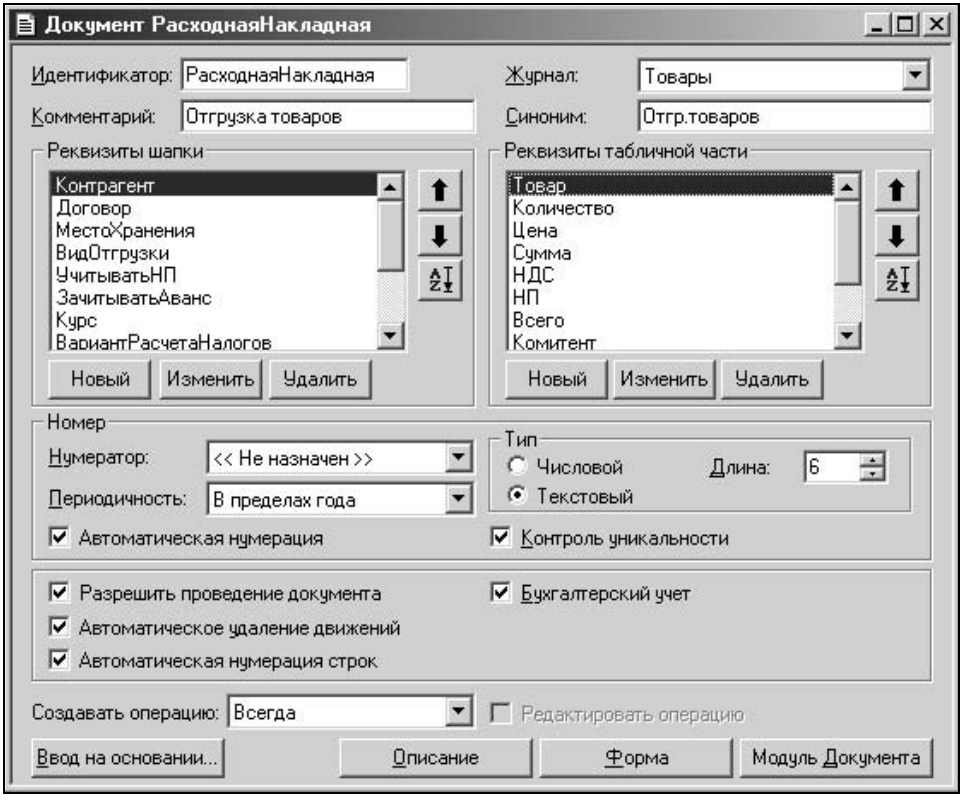


Рис. 1.6. Диалоговое окно настройки документа "РасходнаяНакладная"

Описания всех объектов метаданных, составленных подобным образом, отличаются количеством и последовательностью параметров. Рассмотрим соответствие параметров, заданных в Конфигураторе, параметрам в приведенном фрагменте на примере реквизита "Всего" (рис. 1.7).

Рис. 1.7. Диалоговое окно **Свойства Реквизита** на вкладке **Дополнительные**

Из фрагмента видно, что параметры реквизита, представленные на вкладке **Общие**, совпадают с параметрами реквизитов справочников на данной вкладке. Вкладка **Дополнительные** имеет характерный для табличной части параметр **Итог по колонке**.

Структура файла 1Cv7.dd

На основе файла конфигурации 1Cv7.md формируется словарь базы данных 1Cv7.dd. База данных состоит из совокупности файлов формата Dbase. Отсутствие файла справочника "Валюты" в каталоге базы данных не вызывает сообщение об ошибке. Программа создаст новый файл и его индексы для пустого справочника на основании словаря базы данных. Отсутствие описания справочника в словаре вызовет фатальную ошибку при обращении к справочнику в процессе работы. Рассмотрим фрагмент файла, в котором хранится описание справочника "Валюты". Он имеет следующую структуру:

#==TABLE no 11 : Справочник Валюты				
# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=SC104	Справочник Валюты	A	SC104	1
#-----Fields-------				
# Name	Descr	Type	Length	Precision
F=ID	ID object	C	9	0
F=CODE	object code	C	3	0

F=DESCR	object description	C	25	0
F=ISMARK	Flag Object is Marke	C	1	0
F=VERSTAMP	Version stamp	C	6	0
F=SP98	(Р)ПолнНаименование	C	50	0
F=SP101	(Р)ИмяФайлаПрописи	C	12	0
#----Indexes------				
# Name	Descr	Unique	Indexed fields	DBName
I=IDD	of ID	0	ID	IDD
I=CODE	of CODE	0	CODE(UPPER)	CODE
I=DESCR	of DESCR	0	DESCR(UPPER)	DESCR

Приведенный выше фрагмент условно делится на три части:

- описание таблицы — TABLE;
- описание полей — Fields;
- описание индексов — Indexes.

Описание таблицы приводится в графах с наименованиями:

- Name — имя файла (+.dbf);
- Descr — наименование объекта метаданных (напомним, что связь с файлом конфигурации осуществляется по номеру);
- Type(A/S/U) — тип таблицы (зависит от типа используемой конфигурации);
- DBTableName — имя таблицы (важно для SQL-сервера);
- ReUsable — признак повторного использования.

Из описания видно, что файл справочника называется sc104.dbf. Имена справочников состоят из префикса "sc" и номера объекта метаданных в файле конфигурации 1Cv7.md, а номера документов — соответственно из префикса "dh" и номера. Описание полей приводится в графах с наименованиями:

- Name — имя поля;
- Descr — краткое описание назначения поля;
- Type — тип значения поля (C, N, D);
- Length — длина;
- Precision — точность.

В части описания полей, в которых задаются стандартные для Dbase файлов тип, длина, точность, представляют интерес поле ID и способ задания имен

полей. ID — внутренний код элемента, он не доступен пользователю для редактирования и предназначен для сохранения ссылок при изменении кода. Все ссылки на элементы справочников, документов, перечислений и т. д. делаются с помощью ID. Поля CODE и Descr являются атрибутами справочника, соответственно — Код и Наименование. Реквизитам присваиваются имена, состоящие из префикса "sp" и номера, взятого из файла конфигурации 1Cv7.md.

Следует обратить особое внимание на отсутствие в приведенном фрагменте реквизитов "Курс" и "Кратность".

Внимание

Значения периодических реквизитов не хранятся в основных файлах справочников и документов. Они хранятся в файле констант 1sconst.dbf. Способ организации ссылок в общем файле констант будет рассмотрен далее.

Описание индексов приводится в графах с наименованиями:

- Name — имя индекса;
- Descr — краткое описание индекса;
- Unique — флаг уникальности индекса;
- Indexed fields — строка задания индексируемых полей.

Задание ключевых полей в индексном файле, имя которого совпадает с именем таблицы, имеет одну особенность: индексация по полям задается на верхнем регистре.

Рассмотрим фрагмент подчиненного иерархического файла-справочника "Договоры".

#==TABLE no 16	: Справочник Договоры			
# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=SC112	Справочник Договоры	A	SC112	1
#-----Fields-------				
# Name	Descr	Type	Length	Precision
F=ID	ID object	C	9	0
F=PARENTID	ID parent obj	C	9	0
F=CODE	object code	C	6	0
F=DESCR	object description	C	85	0
F=PARENTEXT	Parent in other tabl	C	9	0
F=ISFOLDER	Flag - Is Line - Fol		N	1
F=ISMARK	Flag Object is Marke	C	1	0

F=VERSTAMP	Version stamp	C	6	0
F=SP13387	(P)ДатаВозникновения	D	8	0
F=SP13386	(P)ДатаПогашенияОбяз	D	8	0
F=SP15926	(P)ВалютаДоговора	C	9	0
F=SP15927	(P)ОплатаДоговора	N	2	0
F=SP22916	(P)ДатаНачалаНачисле	D	8	0
F=SP22917	(P)ДатаПрекращенияНа	D	8	0
F=SP22918	(P)СтавкаШтрафныхСан	N	6	2
F=SP22919	(P)ВременнаяЕдиницаР	C	25	0
F=SP27276	(P)АвтоОбработкаНДС	N	2	0

#----Indexes------

# Name	Descr	Uniq	Indexed fields	DBName
I=IDD	of ID	0	ID	IDD
I=PCODE	of PARENT and	0	PARENTEXT,PARENTID,ISFOLDER, CODE(UPPER)	PCODE
I=PDESCR	of PARENT and	0	PARENTEXT,PARENTID,ISFOLDER, DESCR(UPPER)	PDESCR
I=CODE	of CODE	0	CODE(UPPER)	CODE
I=DESCR	of DESCR	0	DESCR(UPPER)	DESCR
I=VI13387	VI13387	0	SP13387,DESCR(UPPER)	VI13387
I=VIP13387	VIP13387	0	PARENTEXT,PARENTID,ISFOLDER, SP13387,DESCR(UPPER)	VIP13387
I=VI13386	VI13386	0	SP13386,DESCR(UPPER)	VI13386
I=VIP13386	VIP13386	0	PARENTEXT,PARENTID,ISFOLDER, SP13386,DESCR(UPPER)	VIP13386

Поле PARENT задает ссылку на элемент родитель в иерархии. Поле ISFOLDER определяет флаг:

- 0 — элемент не является группой;
- 1 — элемент является группой.

Поле PARENTEXT задает ссылку на элемент справочника родителя, которому подчинен данный элемент.

Документ, имеющий табличную часть, хранится в двух файлах. Реквизиты шапки хранятся в файле с префиксом "dh", реквизиты табличной части хранятся в файле с префиксом "dt". Следовательно, в словаре базы данных

присутствуют два фрагмента, описывающие такой документ. Описание структуры документа в словаре базы данных рассмотрим на примере фрагмента описания структуры документа — "Расходная накладная". Фрагмент, описывающий файл шапки документа, имеет вид:

#==TABLE no 115 : Документ РасходнаяНакладная

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=DH294	Документ РасходнаяНакладная	A	DH294	1
#-----Fields-------				
# Name	Descr	Type	Length	Precision
F=IDDOC	ID Document's	C	9	0
F=SP277	(P)Контрагент	C	9	0
F=SP278	(P)Договор	C	9	0
F=SP279	(P)МестоХранения	C	9	0
F=SP280	(P)ВидОтгрузки	N	2	0
F=SP281	(P)УчитыватьНП	N	2	0
F=SP282	(P)ЗачитыватьАванс	N	2	0
F=SP16096	(P)Курс	N	11	4
F=SP16097	(P)ВариантРасчетаНал	C	9	0
F=SP16098	(P)РасчетныйСчет	C	9	0
F=SP16099	(P)ВерсияОбъекта	C	8	0
F=SP27347	(P)НДСвключатьВСтоим	N	2	0
F=SP27348	(P)ДокументПоступлен	C	9	0
F=SP287	(P)НДС	N	16	2
F=SP290	(P)Всего	N	16	2
#----Indexes------				
# Name	Descr	Unique	Indexed fields	DBName
I=ID	of IDDOC	0	IDDOC	ID

Фрагмент, описывающий файл табличной части документа, приведен ниже.

#==TABLE no 116 : Документ (Мн.ч.) РасходнаяНакладная

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=DT294	Документ (Мн.ч.) РасходнаяНакл	A	DT294	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=IDDOC	ID Document's	C	9	0
F=LINENO	LineNo	N	4	0
F=SP283	(P)Товар	C	9	0
F=SP284	(P)Количество	N	15	3
F=SP285	(P)Цена	N	16	2
F=SP286	(P)Сумма	N	16	2
F=SP287	(P)НДС	N	16	2
F=SP289	(P)НП	N	16	2
F=SP290	(P)Всего	N	16	2
F=SP291	(P)Комитент	C	9	0
F=SP292	(P)ДоговорКомиссии	C	9	0
F=SP27350	(P)ГТД	C	9	0

#----Indexes------

# Name	Descr	Uniq	Indexed fields	DBName
I=IDLINE	of IDDOC+LineN	0	IDDOC,LINENO	IDLINE

В поле LINENO хранится номер строки табличной части документа.

Внимание

Иногда складывается впечатление, что дата, время и номер документа хранятся в файле документа. На самом деле данные атрибуты документа хранятся в журнале.

Основные файлы базы данных

Кроме справочников и документов, которые необходимы в любой конфигурации, в словаре описывается ряд таблиц метаданных, а также системные таблицы. Информация журналов и констант ведется в общих файлах, в отличие от справочников и документов. Поддержка корректной работы в локальной сети, распределение пользователей, а также использование системной информации осуществляется с помощью файлов, описание которых представлено в начале словаря базы данных.

Общие обязательные таблицы для всех конфигураций

Семь первых таблиц базы данных словаря являются обязательными для всех конфигураций независимо от используемых библиотек. В конце словаря базы данных представлены специфические таблицы, которые характерны для определенных предметно-ориентированных конфигураций, использующих метаданные типа "Операция", "Регистр" или "Расчет". Использование свойств и методов этих объектов метаданных связано с наличием специальных библиотек Account.dll, Trade.dll и Salary.dll. При проектировании конфигурации необходимо учитывать характерные особенности каждой компоненты. В данной главе рассматриваются особенности данных компонент с точки зрения построения файловой системы.

Структура файла 1susers.dbf

В данном файле фиксируются активные пользователи конфигурации в сети.

#==TABLE no 0	: Соединений			
# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SUSERS	Соединений	A	1SUSERS	1
#-----Fields-------				
# Name	Descr	Type	Length	Precision
F=USRSCNT	Number of users	N	4	0
F=NETCHGCN	Count of changes	N	10	0

В фрагменте описания файла в словаре базы данных реквизит USRSCNT ("Номер пользователя") используется в режиме — *Монитор* для связи с файлом описания пользователей users.usr.

Внимание

Файл users.usr создается при создании первого пользователя и находится в каталоге конфигурации USRDEF. В конфигурации, в которой не используется список пользователей, данного файла и соответствующего каталога нет.

Структура файла 1ssystem.dbf

Данный файл системного назначения. В нем хранится информация о текущей дате и времени, а также такой важный параметр, как "точка актуальности" (ТА). В поле FLAGS хранятся флаги текущего состояния системы.

#==TABLE no 1	: Системная			
# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SSYSTEM	Системная	A	1SSYSTEM	1
#-----Fields-------				
# Name	Descr	Type	Length	Precision
F=CURDATE	Date of TM	D	8	0
F=CURTIME	Time of TM	C	6	0
F=EVENTIDTA	ID Event On TA	C	9	0
F=DBSIGN	DB Sign	C	3	0
F=DBSETUUID	UUID of DB set	C	36	0
F=SNAPSHPER	Snap Shot Period	C	1	0
F=ACCDATE	Date of Account Tota	D	8	0
F=FLAGS	FLAGS	N	10	0

Структура файла 1sconst.dbf

В фале 1sconst.dbf хранятся значения констант и значения периодических реквизитов документов, поэтому структура файла, как показано ниже, достаточно сложная.

#==TABLE no 2	: Константы			
# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SCONST	Константы	A	1SCONST	1
#-----Fields-------				
# Name	Descr	Type	Length	Precision
F=OBJID	ID obj(0-cons)	C	9	0
F=ID	ID parameter	C	4	0
F=DATE	Fix date	D	8	0
F=PARTNO	no of part	N	3	0
F=VALUE	valume	C	23	0
F=DOCID	ID Document	C	9	0
F=TIME	Time	C	6	0
F=ACTNO	Action No	N	6	0
F=LINENO	LineNo	N	4	0
F=TVALUE		C	3	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=IDD		0	ID,OBJID,DATE,TIME,DOCID,PARTNO	IDD
I=DOC		0	DOCID,ACTNO,PARTNO	DOC

Назначение полей данной таблицы различное для констант и справочников. Для констант в поле ID хранится номер константы как объекта метаданных, под которым он описан в файле 1Cv7.md. Поле OBJID используется в периодических константах для ведения истории значения константы. Для периодических реквизитов справочников в поле ID также хранится номер реквизита справочника как объекта метаданных, под которым он описан в файле 1Cv7.md, Однако в поле OBJID хранится внутренний идентификатор элемента справочника. Если же значение реквизита было изменено документом, то в поле DOCID хранится внутренний идентификатор этого документа. Рассмотрим фрагмент файла, в котором хранится изменение истории значений периодических реквизитов справочника "Основные средства" с помощью документа "Ввод в эксплуатацию" и в диалоговом окне справочника (см. рис. 1.8).

1SCONST : таблица

OBJID	ID	DATE	PARTNO	VALUE	DOCID	TIME	ACTNO	LINENO	TVALUE
9	8TW	28.04.2003	0	2	1P3	65ZOM8	1	0	
9	9HT	28.04.2003	0	1	1P3	65ZOM8	2	0	
9	AQL	28.04.2003	0	0	1P3	65ZOM8	3	0	
9	AQN	28.04.2003	0	0.00	1P3	65ZOM8	4	0	
9	AQO	28.04.2003	0	0	1P3	65ZOM8	5	0	
9	8TX	28.04.2003	0	AQU	1P3	65ZOM8	6	0	
A	8TW	28.04.2003	0	2	1P7	661TS0	1	0	
A	9HT	28.04.2003	0	1	1P7	661TS0	2	0	
A	AQL	28.04.2003	0	0	1P7	661TS0	3	0	
A	AQN	28.04.2003	0	0.00	1P7	661TS0	4	0	
A	AQO	28.04.2003	0	0	1P7	661TS0	5	0	
A	8TX	28.04.2003	0	AQU	1P7	661TS0	6	0	
B	8TW	20.06.2003	0	1	1Q6	BTVSG0	1	0	
B	9HT	20.06.2003	0	1	1Q6	BTVSG0	2	0	
B	AQL	20.06.2003	0	0	1Q6	BTVSG0	3	0	
0	KIN		0	111111111111	0	0	0	0	
9	8TW	28.04.2003	0	2	0	0	0	0	
9	9HT	28.04.2003	0	1	0	0	0	0	
9	AQL	28.04.2003	0	0	0	0	0	0	
A	8TW	01.01.1980	0	1	0	0	0	0	

Запись: 1163 из 1949

Рис. 1.8. Таблица 1SCONST

На рис. 1.8 видно, что значение реквизита VALUE = 1. Оно задано 01.01.1980, с ID = 8TW ("Подразделение" в справочнике "Основные средства")

элемента справочника с номером OBJID = A. Оно было изменено документом с DOCID = 1P7 28.04.2003 на значение VALUE = 2.

Структура файла 1sjourn.dbf.

Все журналы хранятся в файле 1sjourn.dbf. Рассмотрим фрагмент словаря базы данных, описывающий структуру данного файла.

#==TABLE no 3 : Журналы

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SJOURN	Журналы	A	1SJOURN	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=IDJOURNAL	ID of Journal	C	4	0
F=IDDOC	ID Document	C	9	0
F=IDDOCDEF	ID Def Document	C	4	0
F=APPCODE	Application code	N	3	0
F=DATE	date	D	8	0
F=TIME	Time	C	6	0
F=DNPREFIX	Prefix Document No	C	18	0
F=DOCNO	Document No	C	10	0
F=CLOSED	Flag Document is Clo	N	1	0
F=ISMARK	Flag Document is Mar	C	1	0
F=ACTCNT	Action counter	C	6	0
F=VERSTAMP	Version stamp	C	6	0

#----Indexes------

# Name	Descr	Uniq	Index fields	DBName
I=IDDOC	Id Doc	0	IDDOC	IDDOC
I=ACDATETIM	Date+Time+ID	0	DATE,TIME,IDDOC	ACDATETIME
I=DOCNO	Prefix+No	0	DNPREFIX,DOCNO(UPPER)	DOCNO
I=DOCTYPE	Type+Date+Time	0	IDDOCDEF,DATE,TIME, IDDOC	DOCTYPE
I=JOURNAL	Journal+Date+T	0	IDJOURNAL,DATE,TIME, IDDOC	JOURNAL

Необходимо напомнить, что все поля с именами, начинающимися символами "ID", содержат внутренние коды записей, что позволяет поддерживать

целостность информации о ссылках. Для однозначной идентификации документа в конкретном журнале служат поля:

- IDJOURNAL — идентификатор журнала;
- IDDOC — идентификатор документа;
- IDDOCDEF — идентификатор описания документа.

Поле APPCODE носит скорее служебный характер (см. рис. 1.9). Информация о том, был ли создан документ непосредственно в журнале, или введен на основании другого документа, или создан путем обработки некоторых данных, необходима для работы системы в определенных режимах. Значения этого поля могут быть полезны для всякого рода "разборок". Например:

- 4 — документ был создан какой-либо обработкой;
- 20 документ был создан непосредственно в журнале как автономный;
- 52 документ был создан на основании другого документа.

1SJOURN : таблица

IDJOURNAL	IDDOC	IDDOCDEF	APPCODE	DATE	TIME	DNPREFIX	DOCNO	CLOSED	ISMARK	ACTCNT	VERSTAMP
9O2	270	9NP	20	22.07.2003	9D8RWG	125172003	7-655	4		0	0
9O2	272	9NP	20	23.07.2003	9HK61C	125172003	7-656	4		0	0
9O2	27K	9NP	20	25.07.2003	98HY7K	125172003	7-659	4		0	0
9O2	2BP	9NP	20	28.07.2003	9I13LS	125172003	7-661	4		0	0
9O2	2BU	9NP	20	28.07.2003	BCTPYO	125172003	7-663	4		0	0
9O2	2C3	9NP	20	30.07.2003	AG6B5C	125172003	7-665	4		0	0
9O2	2C4	9NP	20	31.07.2003	72XDSW	125172003	7-666	4		0	0
9O2	27L	9NP	20	01.08.2003	5ON41S	125172003	8-133	4		0	0
9O2	2C5	9NP	20	01.08.2003	5UM38W	125172003	8-134	4		0	0
9O2	282	9NP	4	01.08.2003	759EHS	125172003	8-0001	0		0	0
9O2	283	9NP	4	01.08.2003	75BJNK	125172003	8-0002	0		0	0
9O2	284	9NP	4	01.08.2003	75DOTC	125172003	8-0003	0		0	0
9O2	285	9NP	4	01.08.2003	75FTZ4	125172003	8-0004	0		0	0
9O2	286	9NP	4	01.08.2003	75HZ4W	125172003	8-0005	0		0	0
9O2	287	9NP	4	01.08.2003	75K4AO	125172003	8-0006	0		0	0
9O2	288	9NP	4	01.08.2003	75M9GG	125172003	8-0007	0		0	0
9O2	289	9NP	4	01.08.2003	75OEM8	125172003	8-0008	0		0	0
9O2	28A	9NP	4	01.08.2003	75QJS0	125172003	8-0009	0		0	0
9O2	28B	9NP	4	01.08.2003	75SOXS	125172003	8-0010	0		0	0
9O2	28C	9NP	4	01.08.2003	75UU3K	125172003	8-0011	0		0	0
9O2	28D	9NP	4	01.08.2003	75WZ9C	125172003	8-0012	0		0	0
9O2	28E	9NP	4	01.08.2003	75Z4F4	125172003	8-0013	0		0	0
9O2	28F	9NP	4	01.08.2003	7619KW	125172003	8-0014	0		0	0
9O2	28G	9NP	4	01.08.2003	763EQO	125172003	8-0015	0		0	0
9O2	28H	9NP	20	01.08.2003	765JWG	125172003	8-0016	4		0	0
9O2	28I	9NP	4	01.08.2003	767P28	125172003	8-0017	0		0	0

Запись: 1 из 3191

Рис. 1.9. Таблица 1SJOURN

Структура файла 1scrdoc.dbf

Все документы, введенные на основании других документов, участвуют в так называемой структуре подчиненности. Для отражения полной структуры подчиненности документов используется файл 1scrdoc.dbf. Рассмотрим фрагмент словаря базы данных, описывающий структуру данного файла.

#==TABLE no 4 : Ссылки документов

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SCRDOC	Ссылки документов	A	1SCRDOC	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=MDID	Md ID of select	C	4	0
F=PARENTVAL	Parent Value	C	23	0
F=CHILDDATE	Child date	D	8	0
F=CHILDTIME	Child Time	C	6	0
F=CHILDID	Child Doc ID	C	9	0
F=FLAGS	Flags of refers	N	2	0

#----Indexes------

# Name	Descr	Uniq	Indexed fields	DBName
I=CHILD	Child Referenc	0	CHILDID,MDID,PARENTVAL	CHILD
I=PARENT	Parent Referen	0	MDID,PARENTVAL,CHILDDATE, CHILDTIME,CHILDID	PARENT

Заданное подобным образом индексирование позволяет быстро восстанавливать схему подчиненности документа.

Структура файла 1sdnlock.dbf

Данный файл используется при работе с распределенной базой данных. Фрагмент его приведен ниже.

#==TABLE no 5 : Номеров документов

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SDNLOCK	Номеров документов	A	1SDNLOCK	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=DNPREFIX	Prefix object	C	28	0
F=DOCNO	Object No	C	10	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=DOCNO	Prefix+No	0	DNPREFIX,DOCNO	DOCNO

Файлы компоненты "Расчет"

Компонента "Расчет" предназначена для ведения учета объектов с большим количеством разнообразных операций, так называемых "расчетов". Поскольку виды расчетов могут задаваться пользователем, необходимы специфические журналы расчетов. Журналы расчетов хранятся в отдельных файлах подобно документам и справочникам. Имена журналов состоят из префикса "cj" и номера объекта метаданных. Номера объектов метаданных находятся в файле конфигурации 1Cv7.md. Рассмотрим журнал расчета заработной платы. Описание журнала представлено в файле 1Cv7.md следующим фрагментом:

{"CJ",

{"447","Зарплата","Журнал расчета заработной платы","Журнал зарплата", "16","2","2450419",

{"CJParams",

{"448","Дни","Отработанные дни","Дни","N","6","2","0","0","0"},

{"449","Часы","Отработанные часы","Часы","N","7","2","0","0","0"},

{"1089","НомерСтрокиДокумента","Номер строки документа","Номер строки документа","N","4","0","0","1","0"},

{"456","СтрокаИсправления","Номер строки документа-исправления","Строка документа-исправления","N","3","0","0","0","0"}},"15","2","1",

{"FF","202"},

{"Form",

{"2703","ФормаСписка","",""},

{"2707","ПоСотруднику","Открывается из спр. Сотрудники",""}},"2703","2703"},

{"757","Дополнительный","Расчет дополнительной заработной платы","Журнал дополнительной зарплаты","16","2","2450419",

{"CJParams",

{"752","Дни","Дни","Дни","N","6","2","0","0","0"},

{"753","Часы","Часы","Часы","N","7","2","0","0","0"},

{"1236","НомерСтрокиДокумента","Номер строки документа","Номер строки документа","N","4","0","0","0","0"}},"15","2","0",

{"FF","202"},

{"Form",

{"755","ФормаСписка","",""}},"755","755"},

Соответствующее описание данного журнала в словаре базы данных представлено фрагментом:

#==TABLE no 123 : Журнал расчетов Зарплата

# Name	Descr	Type[A/S/U]	DBTable-Name	ReUsable
T=CJ447	Журнал расчетов Зарплата	A	CJ447	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=IDDOC		C	9	0
F=IDS		C	9	0
F=IDALG		C	4	0
F=ORDER		N	3	0
F=RESULT	Result	N	15	2
F=DATEB	date1	D	8	0
F=DATEE	date2	D	8	0
F=PERIOD		C	9	0
F=RECALC		N	3	0
F=ID		C	9	0
F=DP		N	1	0
F=IDPARDOC		C	9	0
F=IDRECALC		C	9	0
F=FF202	(P)ОсновнойЭлемент	C	9	0
F=SP448	(P)Дни	N	7	2
F=SP449	(P)Часы	N	8	2
F=SP1089	(P)НомерСтрокиДокуме	N	5	0
F=SP456	(P)СтрокаИсправления	N	4	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=IDDOC+PER		0	IDDOC,PERIOD,IDS,ORDER	IDDOC+PERIO
I=PERIOD+ID		0	PERIOD,IDS,ORDER,DATEB	PERIOD+IDS+
I=IDS+PERIO		0	IDS,PERIOD,ORDER,DATEB	IDS+PERIOD+
I=ID		0	ID	ID
I=IDS+DATEE		0	IDS,DATEE,ID	IDS+DATEE+I
I=DATEE+ID		0	DATEE,ID	DATEE+ID

I=IDPARDOC		0	IDPARDOC	IDPARDOC
I=IDRECALC		0	IDRECALC	IDRECALC
I=FF202	FF202	0	FF202,PERIOD,IDS,ORDER,DATEB	FF202

В представленном фрагменте особый интерес представляет индексация файла. Кроме индексации по обязательным полям, используется индексация по необязательному полю FF202 ("ОсновнойЭлемент"). Другими словами, изменение структуры справочника "Сотрудники" в части реквизита ОсновнойЭлемент может повлечь нарушение целостности базы данных.

Файлы компоненты "Оперативный учет"

Компонента "Оперативный учет" предназначена для ведения учета накопительных объектов, так называемых "регистров". Регистры хранятся в отдельных файлах, только содержимое каждого регистра хранится в двух файлах. Остатки или обороты в зависимости от заданного в конфигураторе вида хранятся в файлах, имена которых состоят из префикса "rg" и номера объекта метаданных. Расшифровки движений документов по регистру хранятся в файлах, имена которых состоят из префикса "ra" и номера объекта метаданных. Номера объектов метаданных находятся в файле конфигурации 1Cv7.md. Рассмотрим регистр Банк. Описание регистра представлено в файле 1Cv7.md следующим фрагментом:

{"Registers",

{"639","Банк","Остатки средств на банковских счетах","","0"," ","0","1",

{"Props",

{"4066","Фирма","","","B","0","0","4014","0","0","0","0"},

{"641","БанковскийСчет","","Банковский счет","B","0","0","1710","0","0","0","0"}},

{"Figures",

{"644","СуммаВал","Остаток средств в валюте счета","Сумма (вал.)","N","15","2","0","0","0"},

{"642","СуммаУпр","Остаток средств по упр. учету","Сумма (упр.)","N","15","2","0","0","0"},

{"643","СуммаРуб","Сумма по бухгалтерскому учету","Сумма (руб.)","N","15","2","0","0","0"}},

{"Flds",

{"686","КодОперации","","Код операции","E","0","0","345","0","0","0"},

{"4274","ДвижениеДенежныхСредств","","Движение денежных средств","B","0","0","4264","0","0","0"}}},

Соответствующее описание накопительной части данного регистра в словаре базы данных представлено фрагментом:

#==TABLE no 144 : Регистр Банк

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=RG639	Регистр Банк	A	RG639	1
#-----Fields-------				
# Name	Descr	Type	Length	Precision
F=PERIOD	Period Registr	D	8	0
F=SP4066	(P)Фирма	C	9	0
F=SP641	(P)БанковскийСчет	C	9	0
F=SP644	(P)СуммаВал	N	16	2
F=SP642	(P)СуммаУпр	N	16	2
F=SP643	(P)СуммаРуб	N	16	2
#----Indexes------				
# Name	Descr	Unique	Indexed fields	DBName
I=PROP	PERIOD+PROP	0	PERIOD,SP4066,SP641	PROP

Следует обратить особое внимание на то, что суммы в поле SP642 хранятся в валюте, назначенной для управленческого учета, в данном случае в USD (рис. 1.10).

RG639 : таблица

PERIOD	SP4066	SP641	SP644	SP642	SP643
01.04.2002	1	4	35100	35100	1092283,92
01.04.2002	1	U	414843,28	13420,55	414843,28
01.04.2002	2	4	-2080	-2080	-64998,19
01.04.2002	2	2	75000	2480,03	75000
01.04.2002	2	U	2000	64,7	2000
01.05.2002	1	4	33900	33900	1054940,88
01.05.2002	1	U	352119,28	11391,37	352119,28
01.05.2002	2	2	-186440	-5976,27	-186440
01.05.2002	2	4	-2080	-2080	-64998,19
01.05.2002	2	U	2000	64,7	2000
01.06.2002	1	4	33900	33900	1054940,88
01.06.2002	1	U	352119,28	11391,37	352119,28
01.06.2002	2	2	-186440	-5976,27	-186440
01.06.2002	2	4	-2080	-2080	-64998,19

Запись: 33 из 90

Рис. 1.10. Таблица RG639

RA639 : таблица

IDDOC	LINENO	ACTNO	DEBKRED	SP4066	SP641	SP644	SP642	SP643	SP686	SP4274
31	0	5	0	1	4	36000	36000	1120291,2	2XZ	0
31	0	6	0	1	U	150000	5000	150000	2XY	0
32	0	65	0	2	4	300	300	9335,76	2XZ	0
32	0	66	0	2	2	100000	3333,33	100000	2XY	0
6E	0	36	0	1	U	467566,5	15000	467566,5	1PS	H
54	0	68	0	2	2	4000	128,32	4000	1MN	2
55	0	107	0	2	4	0	0	15,57	1G7	8
55	0	108	1	2	4	10000	10000	311711	3S1	8
1W	0	107	0	1	U	0	-19,89	0	1G7	2
1W	0	108	1	1	U	65400	2098,1	65400	1ML	2
3Q	0	48	0	2	2	23000	737,86	23000	1MN	9
53	0	132	0	1	U	0	-41,36	0	1G7	8
5M	0	72	0	1	U	52881,4	1696,49	52881,4	1MN	2
28	0	146	0	1	U	0	-5,25	0	1G7	0
28	0	147	1	1	U	21000	674,12	21000	1ML	0
56	0	104	0	2	2	0	-50,22	0	1G7	8
56	0	105	1	2	2	52000	1669,26	52000	1ML	8
5Z	1	68	0	2	4	7620	7620	237361,48	3S0	2
5Q	1	151	0	1	U	0	-8	0	1G7	8
5Q	1	152	1	1	U	31245,63	1002,82	31245,63	1ML	8
22	1	101	0	1	4	0	0	70,65	1G7	0
22	1	102	1	1	4	900	900	28077,93	3S1	0

Запись: 24 из 33

Рис. 1.11. Таблица RA639

Соответствующее описание расшифровки движений данного регистра в словаре базы данных представлено фрагментом:

#==TABLE no 145 : Регистр (Дв.) Банк

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=RA639	Регистр (Дв.) Банк	A	RA639	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=IDDOC	ID Document's	C	9	0
F=LINENO	LineNo	N	4	0
F=ACTNO	Action No	N	6	0
F=DEBKRED	Flag Debet/Kredit	N	1	0
F=SP4066	(P)Фирма	C	9	0
F=SP641	(P)БанковскийСчет	C	9	0
F=SP644	(P)СуммаВал	N	16	2
F=SP642	(P)СуммаУпр	N	16	2
F=SP643	(P)СуммаРуб	N	16	2

F=SP686	(P)КодОперации	C	9	0
F=SP4274	(P)ДвижениеДенежныхС	C	9	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=IDLINE	of IDDOC+LineN	0	IDDOC,LINENO,ACTNO	IDLINE

Из приведенного на рис. 1.11 фрагмента таблицы RA639 видно, что остаток в USD 35100 (рис. 1.10) сложился из дебетового оборота 36000 (DEBKRED=0) минус кредитовый оборот 900 (DEBKRED=1).

Структура файла 1sstream

Для компоненты "Оперативный учет" характерно ведение последовательности документов. Последовательности ведутся в файле 1sstream.dbf. Описание последовательностей представлено в файле 1Cv7.md следующим фрагментом:

{"Document Streams",

{"1946","ОсновнаяПоследовательность","","Основная последовательность","0","0",

{"Registers","4314","4335","328","351","635","639","2964","405","438","464","4667","4674","4480"},

{"Documents","1582","1611","1656","1684","1774","1790","2196","2225","3259","2320","2075","3274","3311","1628","3114","2988","2998","3089","2827","3638","5211","4913","5292","3614","4854","4847","3995","4694","3592","2695","2051","4389","3504","5155","2742","2457","6661","4132","6313","6532","6322"}},

{"4757","КнигаПокупок","","Книга покупок","0","0",

{"Registers","3549"},

{"Documents","3592","3504","6661"}},

{"5722","КнигаПродаж","","Книга продаж","0","0",

{"Registers","4343"},

{"Documents","4389","4694","6661"}}},

Во фрагменте приведено описание трех последовательностей: Основная Последовательность, КнигаПокупок, КнигаПродаж.

Соответствующее описание таблицы последовательностей в словаре базы данных представлено фрагментом:

#==TABLE no 176 : Последовательность документов

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SSTREAM	Последовательность документов	A	1SSTREAM	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=ID	Doc Stream ID	C	4	0
F=DATE	Date	D	8	0
F=TIME	Time	C	6	0
F=DOCID	Doc ID	C	9	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=ID	by doc stream	0	ID	ID

Файлы компоненты "Бухгалтерский учет"

Описание плана счетов, операций, проводок, бухгалтерских итогов, остатков, отбора счетов, корректных проводок, типовых операций, отбора проводок по субконто — представлены в конце словаря базы данных.

Структура файла 1saccs.dbf

Один из самых важных файлов бухгалтерии — "Счета". Другими словами, в данном файле хранится план счетов. Структура данного файла показана ниже.

#==TABLE no 191 : Счета

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SACCS	Счета	A	1SACCS	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=ID	Account Id	C	9	0
F=PLANID	Plan Id	C	4	0
F=SCHKOD	Account code(number)	C	11	0
F=SCHIM	Account description	C	25	0
F=SCHV	Flag Currency enable	N	1	0
F=SCHKOL	Flag Amount enable	N	1	0
F=SCHSINGLE	Flag no need corresp	N	1	0
F=ISFOLDER	Flag Have child acco	N	1	0
F=ISMARK	Flag Object is Marke	C	1	0
F=LEVEL	Level of Account	N	3	0

F=MDID	Metadata Id	C	4	0
F=ACTIVE	Flag Active	N	1	0
F=VERSTAMP	Version stamp	C	6	0
F=SC0		C	4	0
F=OSC0		N	1	0
F=FSC0		N	1	0
F=SC1		C	4	0
F=OSC1		N	1	0
F=FSC1		N	1	0
F=SC2		C	4	0
F=OSC2		N	1	0
F=FSC2		N	1	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=ID1	of ID	0	ID	ID1
I=CODE	of CODE	0	PLANID,SCHKOD	CODE
I=SUBACC	of SubAcc	0	LEVEL,PLANID,SCHKOD	SUBACC

Назначение полей следующее:

- PLANID — номер плана счетов, под которым он описан в файле 1Cv7.md;
- ID — "сквозной" внутренний номер счета, независимо от принадлежности к конкретному плану счетов;
- MDID — номер счета, под которым он описан в файле 1Cv7.md;
- SCHKOD — код счета в справочнике;
- SCHIM — наименование счета в справочнике;
- SCHV — флаг валютного учета счета;
- SCHKOL — флаг количественного учета счета;
- SCHSINGLE — флаг принадлежности счета к балансовым счетам;
- ISFOLDER — флаг папки;
- ISMARK — флаг пометки на удаление;
- LEVEL — уровень счета (для субсчетов);
- ACTIVE — флаг вида счета (активный, пассивный или активно-пассивный);

- SC0 — номер вида субконто1, под которым он описан в файле 1Cv7.md;
- OSC0 — флаг учета оборотов по счету для субконто1;
- FSC0 — флаг суммового и количественного учета по субконто1;
- SC1 — номер вида субконто2, под которым он описан в файле 1Cv7.md;
- OSC1 — флаг учета оборотов по счету для субконто2;
- FSC1 — флаг суммового и количественного учета по субконто2;
- SC2 — номер вида субконто3, под которым он описан в файле 1Cv7.md;
- OSC2 — флаг учета оборотов по счету для субконто3;
- FSC2 — флаг суммового и количественного учета по субконто3.

Внимание

Поскольку тип данных в полях внутренних идентификаторов объявлен — "символьный", отображение в стандартных программах просмотра таблиц не совсем понятно (см. рис. 1.12). На самом деле значение номера в поле MDID для счета "01" — 11748, хотя в программе просмотра таблицы он указан — 910.

1SBKTTLC.DBF

	1	2	3	4	5	6	7	8	9	10
1	DATE	ACCDTID	ACCKTID	CURRID	KIND	OB1	OB2	OB3	DTFLAGS	KTFLAGS
2	01.04.2002	K	43	0	1	0.00	5200.00	0.00	2	0
3	01.04.2002	43	4L	0	1	0.00	10000.00	0.00	0	0
4	01.04.2002	3	K	0	1	0.00	5200.00	0.00	0	2
5	01.04.2002	1C	3W	0	1	0.00	360.00	0.00	0	0
6	01.04.2002	1C	3U	0	1	0.00	8000.00	14000.00	0	0
7	01.04.2002	3U	3H	0	1	0.00	1040.00	1820.00	0	0
8	01.01.2003	Z1422B	Z1422D	0	1	530.16	1111.26	1368.18	0	0
9	01.04.2002	1C	3Q	0	1	0.00	320.00	560.00	0	0
10	01.01.2003	Z1422B	Z1422C	0	1	2433.92	4710.64	5955.36	0	0
11	01.04.2002	1C	3T	0	1	0.00	16.00	28.00	0	0
12	01.04.2002	1C	Z1422B	0	1	0.00	1120.00	1960.00	0	0
13	01.04.2002	1C	Z1422C	0	1	0.00	880.00	1630.00	0	0
14	01.04.2002	1C	Z1422D	0	1	0.00	240.00	330.00	0	0
15	01.07.2003	3W	1W	0	1	0.00	5200.00	0.00	0	0
16	01.07.2003	1D	Z1422B	0	1	8071.28	0.00	0.00	0	0
17	01.04.2002	1W	4B	0	1	0.00	0.00	50000.00	0	0
18	01.04.2002	1Z	1W	0	1	0.00	0.00	50000.00	0	0
19	01.04.2003	19	3W	0	1	2290.00	2340.00	2140.00	0	0
20	01.04.2003	3I	4F	0	1	0.00	0.00	3549.50	0	0
21	01.04.2003	4F	3I	0	1	0.00	0.00	14057.00	0	0
22	01.04.2002	5A	1Z	0	1	0.00	0.00	15.82	0	0
23	01.07.2003	Z1422B	Z1422D	0	1	1906.54	0.00	0.00	0	0

1SBKTTLC

Рис. 1.12. Таблица 1SBKTTLC

Структура файла 1soper.dbf

Все операции, сформированные документами, фиксируются в таблице операций 1soper. Если документом были сформированы проводки, они фиксируются еще и в таблице 1sentry. Потеря записи в отдельно взятой таблице неминуемо приведет к нарушению целостности базы данных. В программе существует ряд специальных режимов по восстановлению целостности базы данных, к рассмотрению которых мы вернемся в *гл. "Ремонт"*. Пока рассмотрим структуру файла таблицы операций, фрагмент которого приведен ниже.

#==TABLE no 192 : Операции

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SOPER	Операции	A	1SOPER	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=DOCID	Document Id	C	9	0
F=DATE	date	D	8	0
F=TIME	Time	C	6	0
F=DESCR	Oper description	C	50	0
F=SUM	Oper sum	N	14	2
F=ACTIVE		C	1	0

#----Indexes------

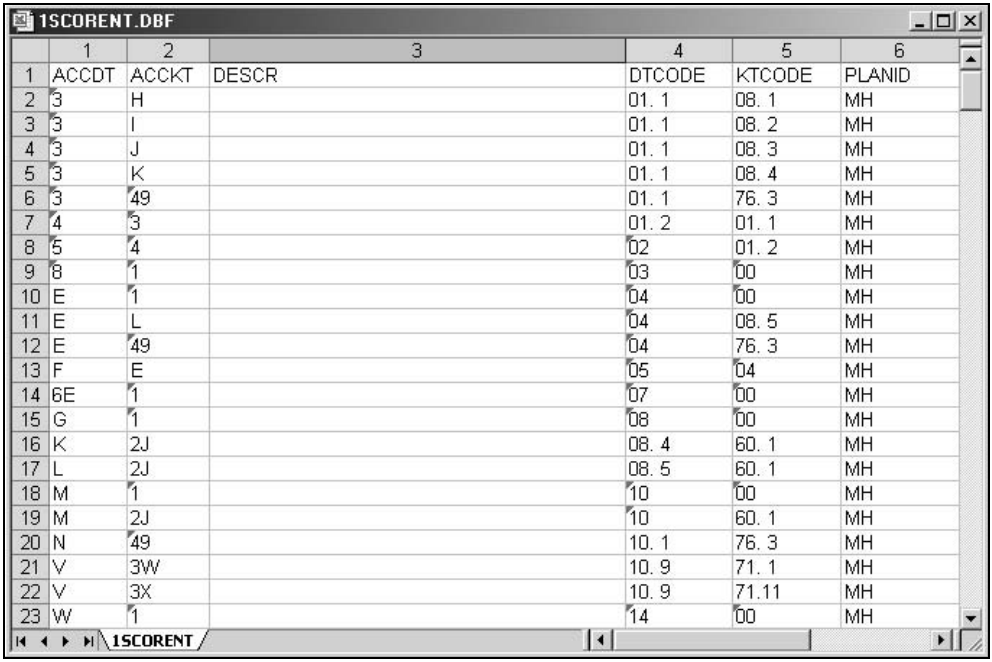
# Name	Descr	Unique	Indexed fields	DBName
I=DOCID1	of ID	0	DOCID	DOCID1
I=DATETIME	ofDateTime	0	DATE,TIME,DOCID	DATETIME
I=SumIdx	of SUM Date Ti	0	SUM,DATE,TIME,DOCID	SumIdx
I=DescrIdx	ofSUMDateTi	0	DESCR(UPPER),DATE,TIME,DOCID	DescrIdx

На рис. 1.13 видно, что журнал операций — таблица с индексами.

Назначение полей таблицы следующее:

- DOCID — внутренний идентификатор документа или ручной операции;
- DATE — дата документа или ручной операции;
- TIME — время документа или ручной операции;
- DESCR — атрибут объекта метаданных Операция "Содержание" с добавленной расшифровкой (атрибут операции формируется при выполнении предопределенной процедуры модуля формы документа `ПриЗаписи()`;

- SUM — атрибут объекта метаданных Операция "СуммаОперации" (атрибут операции формируется при выполнении предопределенной процедуры модуля формы документа `ПриЗаписи()`;
- ACTIVE — признак активности записи.



1SCORENT.DBF

	1	2	3	4	5	6
1	ACCDT	ACCKT	DESCR	DTCODE	KTCODE	PLANID
2	3	H		01. 1	08. 1	MH
3	3	I		01. 1	08. 2	MH
4	3	J		01. 1	08. 3	MH
5	3	K		01. 1	08. 4	MH
6	3	49		01. 1	76. 3	MH
7	4	3		01. 2	01. 1	MH
8	5	4		02	01. 2	MH
9	8	1		03	00	MH
10	E	1		04	00	MH
11	E	L		04	08. 5	MH
12	E	49		04	76. 3	MH
13	F	E		05	04	MH
14	6E	1		07	00	MH
15	G	1		08	00	MH
16	K	2J		08. 4	60. 1	MH
17	L	2J		08. 5	60. 1	MH
18	M	1		10	00	MH
19	M	2J		10	60. 1	MH
20	N	49		10. 1	76. 3	MH
21	V	3W		10. 9	71. 1	MH
22	V	3X		10. 9	71.11	MH
23	W	1		14	00	MH

1SCORENT

Рис. 1.13. Таблица 1SCORENT

Структура файла 1sentry.dbf

Все проводки, сформированные документами, фиксируются в таблице 1sentry. Индексированная таблица 1sentry — это журнал проводок, структура файла которого показана ниже.

#==TABLE no 193 : Проводки

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SENTRY	Проводки	A	1SENTRY	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=DOCID	Document Id	C	9	0
F=NUMBER	Prov number	N	5	0

F=CORNO	Correspond number	N	5	0
F=DATE	date	D	8	0
F=TIME	Time	C	6	0
F=ACCDTID	AccountDt Id	C	9	0
F=ACCKTID	AccountKt Id	C	9	0
F=SUM	Prov sum	N	14	2
F=CURRID	Currency Id	C	9	0
F=CURSUM	Prov currency sum	N	14	2
F=AMOUNT	Prov amount	N	14	3
F=PROVKIND		C	1	0
F=ACTIVE		C	1	0
F=DTFLAGS		N	1	0
F=KTFLAGS		N	1	0
F=DOCLINENO	Document line number	N	4	0
F=SP547	(P)СодержаниеПроводк	C	50	0
F=SP548	(P)НомерЖурнала	C	2	0
F=SP26962	(P)ПервичныйДокумент	C	25	0
F=VDTSC0		C	4	0
F=DTSC0		C	13	0
F=ODTSC0		N	1	0
F=VDTSC1		C	4	0
F=DTSC1		C	13	0
F=ODTSC1		N	1	0
F=VDTSC2		C	4	0
F=DTSC2		C	13	0
F=ODTSC2		N	1	0
F=VKTSC0		C	4	0
F=KTSC0		C	13	0
F=OKTSC0		N	1	0
F=VKTSC1		C	4	0
F=KTSC1		C	13	0
F=OKTSC1		N	1	0

F=VKTSC2			C	4	0
F=KTSC2			C	13	0
F=OKTSC2			N	1	0

#----Indexes------

# Name	Descr	Uniq	Indexed fields	DBName
I=DOCIDNUMB	of ID	0	DOCID,NUMBER,CORNO	DOCIDNUMBER
I=DATETIME	ofDateTime	0	DATE,TIME,DOCID,NUMBER,CORNO	DATETIME
I=SumIdx	of SUM	0	SUM,DATE,TIME,DOCID,NUMBER, CORNO	SumIdx
I=VIA548	VIA548	0	SP548(UPPER=128),DATE,TIME,DOCID, NUMBER,CORNO	VIA548

Поля приведенной выше структуры имеют следующее назначение:

- DOCID — внутренний идентификатор документа или ручной операции;
- DATE — дата документа или ручной операции;
- TIME — время документа или ручной операции;
- NUMBER — порядковый номер проводки, начиная с 0;
- ACCDTID — идентификатор счета дебета (совпадает со значением поля ID в таблице 1saccs);
- ACCKTID — идентификатор счета кредита (совпадает со значением поля ID в таблице 1saccs);
- SUM — сумма проводки;
- CURRID — идентификатор валюты;
- CURSUM — сумма в валюте;
- AMOUNT — количество;
- PROVKIND — вид проводки;
- ACTIVE — признак активности записи;
- DTFLAGS — флаги дебета (количественный учет счета, валютный и т. п.);
- KTFLAGS — флаги кредита (количественный учет счета, валютный и т. п.);
- DOCLINENO — номер строки документа;
- SP547 — атрибут "СодержаниеПроводки";
- SP548 — номер журнала;
- SP26962 — данные первичного документа, графа журнала проводок "Первичный документ";

- VDTSC0 — номер вида субконто1 дебета, под которым он описан в файле 1Cv7.md;
- ODTSC0 — флаг учета оборотов по счету для субконто1 дебета;
- DTSC0 — идентификатор субконто1 дебета;
- VKTSC0 — номер вида субконто1 кредита, под которым он описан в файле 1Cv7.md;
- OKTSC0 — флаг учета оборотов по счету для субконто1 кредита;
- KTSC0 — идентификатор субконто1 кредита;
- VDTSC1 — номер вида субконто2 дебета, под которым он описан в файле 1Cv7.md;
- ODTSC1 — флаг учета оборотов по счету для субконто2 дебета;
- DTSC1 — идентификатор субконто2 дебета;
- VKTSC1 — номер вида субконто2 кредита, под которым он описан в файле 1Cv7.md;
- OKTSC1 — флаг учета оборотов по счету для субконто2 кредита;
- KTSC1 — идентификатор субконто2 кредита;
- VDTSC2 — номер вида субконто3 дебета, под которым он описан в файле 1Cv7.md;
- ODTSC2 — флаг учета оборотов по счету для субконто3 дебета;
- DTSC2 — идентификатор субконто3 дебета;
- VKTSC2 — номер вида субконто3 кредита, под которым он описан в файле 1Cv7.md;
- OKTSC2 — флаг учета оборотов по счету для субконто3 кредита;
- KTSC2 — идентификатор субконто3 кредита.

Структура файла 1sbkttlc.dbf

Для получения оперативных отчетов, программой ведется таблица бухгалтерских итогов 1sbkttlc (см. рис. 1.14). При работе с некоторыми методами объекта метаданных "БухгалтерскиеИтоги" обращение производится к данной таблице. Нарушение в данной таблице приводит к неприятностям при получении ряда отчетов, таких как "Оборотно-сальдовая ведомость", "Оборотно-сальдовая ведомость по счету" и т. п. В программе существует специальный режим по восстановлению бухгалтерских итогов, к рассмотрению которого мы вернемся в *гл. "Рекомендации по сопровождению программы".*

Фрагмент структуры файла, таблицы бухгалтерских отчетов, показан далее.

#==TABLE no 194 : Итоги

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SBKTTLC	Итоги	A	1SBKTTLC	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=DATE	Period	D	8	0
F=ACCDTID	AccountDt Id	C	9	0
F=ACCKTID	AccountKt Id	C	9	0
F=CURRID	Currency Id	C	9	0
F=KIND	Total kind	C	1	0
F=OB1	Total turnover	C	15	0
F=OB2	Total turnover	C	15	0
F=OB3	Total turnover	C	15	0
F=DTFLAGS		N	1	0
F=KTFLAGS		N	1	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=ACCDT1		0	DATE,KIND,ACCDTID,ACCKTID,CURRID	ACCDT1
I=ACCKT1		0	DATE,KIND,ACCKTID,ACCDTID,CURRID	ACCKT1

Назначение полей:

- DATE — период итогов, начало отчетного квартала;
- ACCDTID — идентификатор счета дебета, совпадает со значением поля ID в таблице 1saccs;
- ACCKTID — идентификатор счета кредита, совпадает со значением поля ID в таблице 1saccs;
- CURRID — валюта итогов;
- KIND — вид итогов;
- OB1 — оборот за текущий период;
- OB2 — начальный остаток текущего периода;
- OB3 — конечный остаток текущего периода;
- DTFLAGS — флаги счета дебета;
- KTFLAGS — флаги счета кредита.

1SACCSEL.DBF

	1	2	3	4	5	6	7	8
1	ACCID	DATE	TIME	DOCID	NUMBER	CORNO	DT	KT
2	K	06.05.2002	7579C0	N	0	0	*	
3	G	06.05.2002	7579C0	N	0	0	*	
4	43	06.05.2002	7579C0	N	0	0		*
5	42	06.05.2002	7579C0	N	0	0		*
6	K	06.05.2002	7579C0	N	1	0	*	
7	G	06.05.2002	7579C0	N	1	0	*	
8	43	06.05.2002	7579C0	N	1	0		*
9	42	06.05.2002	7579C0	N	1	0		*
10	43	06.05.2002	7579C0	N	2	0	*	
11	42	06.05.2002	7579C0	N	2	0	*	
12	4L	06.05.2002	7579C0	N	2	0		*
13	43	06.05.2002	7579C0	N	3	0	*	
14	42	06.05.2002	7579C0	N	3	0	*	
15	4L	06.05.2002	7579C0	N	3	0		*
16	3	06.05.2002	7579C0	N	4	0	*	
17	2	06.05.2002	7579C0	N	4	0	*	
18	K	06.05.2002	7579C0	N	4	0		*
19	G	06.05.2002	7579C0	N	4	0		*
20	3	06.05.2002	7579C0	N	5	0	*	
21	2	06.05.2002	7579C0	N	5	0	*	
22	K	06.05.2002	7579C0	N	5	0		*
23	G	06.05.2002	7579C0	N	5	0		*
24	1C	17.05.2002	7579C0	8	0	0	*	
25	3W	17.05.2002	7579C0	8	0	0		*
26	3V	17.05.2002	7579C0	8	0	0		*

1SACCSEL

Рис. 1.14. Таблица 1SACCSEL

Структура файла 1sbkttl.dbf

Для получения оперативных отчетов по субконто, программой ведется таблица бухгалтерских итогов 1sbkttl. При работе с некоторыми методами объекта метаданных "БухгалтерскиеИтоги", для получения итогов по субконто, обращение производится к данной таблице.

#==TABLE no 195 : Остатки

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SBKTTL	Остатки	A	1SBKTTL	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=DATE	Period	D	8	0
F=ACCID	AccountDt Id	C	9	0
F=CURRID	Currency Id	C	9	0
F=KIND	Total kind	C	1	0
F=OBDT1	Total turnover DT	C	15	0
F=OBKT1	Total turnover KT	C	15	0
F=OBDT2	Total turnover DT	C	15	0
F=OBKT2	Total turnover KT	C	15	0
F=OBDT3	Total turnover DT	C	15	0
F=OBKT3	Total turnover KT	C	15	0
F=SD	Saldo	C	15	0
F=FLAGS		N	1	0
F=VSC0		C	4	0
F=SC0		C	13	0
F=OSC0		N	1	0
F=VSC1		C	4	0
F=SC1		C	13	0
F=OSC1		N	1	0
F=VSC2		C	4	0
F=SC2		C	13	0
F=OSC2		N	1	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=ACC1		0	DATE,KIND,ACCID,SC0,SC1,SC2,CURRID	ACC1

Назначение полей:

- DATE — период итогов, начало отчетного квартала;
- ACCID — идентификатор счета дебета, совпадает со значением поля ID в таблице 1saccs;

- CURRID — валюта итогов;
- KIND — вид итогов;
- OBDT1 — обороты по дебету субконто1;
- OBKT1 — обороты по кредиту субконто1;
- OBDT2 — обороты по дебету субконто2;
- OBKT2 — обороты по кредиту субконто2;
- OBDT3 — обороты по дебету субконто3;
- OBKT3 — обороты по кредиту субконто3;
- SD — сальдо;
- FLAGS — флаги;
- VSC0 — номер вида субконто1;
- SC0 — идентификатор субконто1;
- OSC0 — признак учета только оборотов субконто1;
- VSC1 — номер вида субконто2;
- SC1 — идентификатор субконто2;
- OSC1 — признак учета только оборотов субконто2;
- VSC2 — номер вида субконто3;
- SC2 — идентификатор субконто3;
- OSC2 — признак учета только оборотов субконто2.

Структура файла 1scorent.dbf

Перечень корректных проводок хранится в таблице 1scorent (см. рис. 1.15).

#==TABLE no 196 : Корректные проводки

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SCORENT	Корректные проводки	A	1SCORENT	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=ACCDT	Account Dt Id	C	9	0
F=ACCKT	Account Kt Id	C	9	0
F=DESCR	Description	C	50	0
F=DTCODE	Account Dt Code	C	11	0
F=KTCODE	Account Kt Code	C	11	0
F=PLANID	Account Kt Code	C	11	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=DT		0	ACCDT,ACCKT	DT
I=KT		0	ACCKT,ACCDT	KT
I=CODEDT		0	PLANID,DTCODE,KTCODE	CODEDT
I=CODEKT		0	PLANID,KTCODE,DTCODE	CODEKT

Индексы существуют по всем полям. Это позволяет осуществлять оперативный контроль вводимых проводок в соответствии с представленной на рисунке таблицей.

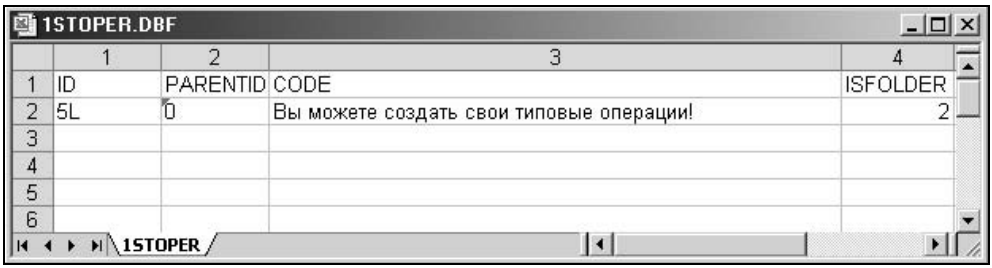


Рис. 1.15. Таблица 1STOPER

Назначение полей:

- ACCDT — идентификатор счета дебета, совпадает со значением поля ID в таблице 1saccs;
- ACCKT — идентификатор счета кредита, совпадает со значением поля ID в таблице 1saccs;
- DESCR — содержание проводки;
- DTCODE — код счета дебета в плане счетов;
- KTCODE — код счета кредита в плане счетов;
- PLANID — номер счета, под которым он описан в файле 1Cv7.md;

Структура файла 1saccsel.dbf

Файл отбора счетов 1saccsel.dbf используется для ускорения выполнения ряда методов объекта метаданных типа "БухгалтерскиеИтоги". Данный файл выполняет роль посредника для доступа к суммовым и количественным показателям бухгалтерских итогов. Описание таблицы отбора счетов в словаре базы данных представлено фрагментом:

#==TABLE no 197 : Отбор Счетов

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SACCSEL	Отбор Счетов	A	1SACCSEL	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=ACCID	Account Id	C	9	0
F=DATE	Date	D	8	0
F=TIME	Time	C	6	0
F=DOCID	Doc Id	C	9	0
F=NUMBER	Prov number	N	5	0
F=CORNO	Correspond number	N	5	0
F=DT		C	1	0
F=KT		C	1	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=MAIN		0	ACCID,DATE,TIME,DOCID,NUMBER,CORNO	MAIN
I=PROV		0	DOCID,NUMBER,CORNO	PROV

Назначение полей:

- ACCID — идентификатор счета, по которому была сформирована проводка;
- DATE — дата проводки;
- TIME — время проводки;
- DOCID — идентификатор документа, которым была сформирована проводка;
- NUMBER — номер проводки в операции;
- CORNO — номер корреспонденции;
- DT — признак дебетового оборота по счету;
- KT — признак кредитового оборота по счету.

Следует обратить особое внимание на количество строк с одним номером проводки. Строк с одним номером проводки может быть больше двух, т. к. строки по счету формируются и для всех вышестоящих уровней счета. Рассмотрим операцию вкладов в уставной капитал на расчетный счет (рис. 1.16).

Журнал операций *

Журнал операций

с 03.01.02 по 03.01.02

Дата	Документ	Номер	Сумма	Содержание			
№ пр	Дебет	Кредит	Сумма	Содержание проводки	Кол-во	Валюта	Вал. Сумма
03.01.02	**Операция 00000002**		**600,000.00**	**вклады в уставной фонд**			
1	75.1	80	50,000.00	Вклад в уставной фонд :О Васильев Ф.С. Васильев Ф.С.			
2	75.1	80	150,000.00	Вклад в уставной фонд :О Элитный банк Элитный банк			
3	51	75.1	25,000.00	: Основной р/с Прочие поступления Васильев Ф.С.			
4	51	75.1	150,000.00	: Основной р/с Прочие поступления Элитный банк			

Рис. 1.16. Операция вкладов в уставной капитал

Во таблице данная операция будет содержать по три строки на каждую проводку, например, в первой проводке: в дебет счетов 75 (ACCID=41), 75.1 (ACCID=4H) и в кредит счета 80 (ACCID=59) (рис. 1.17).

1SACCSEL : таблица

ACCID	DATE	TIME	DOCID	NUMBER	CORNO	DT	KT
4I	03.01.2002	7579C0	EN	0	0	*	
4H	03.01.2002	7579C0	EN	0	0	*	
59	03.01.2002	7579C0	EN	0	0		*
4I	03.01.2002	7579C0	EN	1	0	*	
4H	03.01.2002	7579C0	EN	1	0	*	
59	03.01.2002	7579C0	EN	1	0		*
25	03.01.2002	7579C0	EN	2	0	*	
4I	03.01.2002	7579C0	EN	2	0		*
4H	03.01.2002	7579C0	EN	2	0		*
25	03.01.2002	7579C0	EN	3	0	*	
4I	03.01.2002	7579C0	EN	3	0		*
4H	03.01.2002	7579C0	EN	3	0		*

Запись: 12 из 2144

Рис. 1.17. Таблица 1SACCSEL

Структура файла 1stoper.dbf

В файле 1stoper.dbf хранится справочник типовых операций. На заре развития компоненты "Бухгалтерский учет" данный справочник был весьма актуален. С внедрением налогового учета он практически не используется.

#==TABLE no 198 : Типовые операции (рис. 1.18)

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1STOPER	Типовые операции	A	1STOPER	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=ID	Typ Oper ID	C	9	0
F=PARENTID	ID parent obj	C	9	0
F=CODE	Typ oper Descryption	C	80	0
F=ISFOLDER	Flag - Is Line – Fol	N	1	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=IDD	of ID	0	ID	IDD
I=PCODE	ofPARENTand	0	PARENTID,ISFOLDER,CODE(UPPER)	PCODE
I=CODE	of CODE	0	CODE(UPPER)	CODE

Структура файла 1ssbsel.dbf

Файл отбора проводок по субконто 1ssbsel.dbf используется для ускорения выполнения ряда методов объекта метаданных типа "БухгалтерскиеИтоги". Данный файл выполняет роль посредника для доступа к суммовым и количественным показателям бухгалтерских итогов. Описание таблицы отбора проводок по субконто в словаре базы данных представлено фрагментом:

#==TABLE no 199 :

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=1SSBSEL	Отбор проводок по субконто	A	1SSBSEL	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=SBKINDID	Sb Kind ID of select	C	4	0
F=VAL	Value	C	13	0
F=DATE	Date	D	8	0
F=TIME	Time	C	6	0
F=DOCID	Doc ID	C	9	0
F=NUMBER	Prov number	N	5	0
F=CORNO	Correspond number	N	5	0

```
#----Indexes------
# Name      Descr          Uniq  Indexed fields                          DBName
I=ENTRY     all entries re 0     DOCID,NUMBER,CORNO                     ENTRY
I=VALUE     entrybyvalue   0     SBKINDID,VAL,DATE,TIME,DOCID,          VALUE
                                  NUMBER,CORNO
```

Назначение полей:

- SBKINDID — идентификатор вида субконто, с которым была сформирована проводка;
- VAL — значение субконто, с которым была сформирована проводка;
- DATE — дата проводки;
- TIME — время проводки;
- DOCID — идентификатор документа, которым была сформирована проводка;
- NUMBER — номер проводки в операции;
- CORNO — номер корреспонденции.

Операция вкладов в уставной капитал (рис. 1.16) в файле отбора по субконто будет содержать четыре строки по количеству проводок, т. к. счет 80 не имеет аналитики (рис. 1.18).

1SSBSEL : таблица

SBKINDID	VAL	DATE	TIME	DOCID	NUMBER	CORNO
EE	3B	03.01.2002	7579C0	EN	0	0
EE	44	03.01.2002	7579C0	EN	1	0
EE	3B	03.01.2002	7579C0	EN	2	0
EE	44	03.01.2002	7579C0	EN	3	0

Запись: 4 из 322

Рис. 1.18. Таблица 1SSBSEL

Файлы компоненты "Расчет"

Компонента "Расчет" предназначена для ведения учета объектов с большим количеством разнообразных операций, так называемых "расчетов". Поскольку виды расчетов могут задаваться пользователем, необходимы специфические журналы расчетов. Журналы расчетов хранятся в отдельных файлах подобно документам и справочникам. Имена журналов состоят из префикса "cj" и номера объекта метаданных в файле конфигурации 1Cv7.md. Рассмотрим пример структуры файла.

{"CJ",

{"447","Зарплата","Журнал расчета заработной платы","Журнал-зарплата","16","2","2450419",

{"CJParams",

{"448","Дни","Отработанные дни","Дни","N","6","2","0","0","0"},

{"449","Часы","Отработанные часы","Часы","N","7","2","0","0","0"},

{"1089","НомерСтрокиДокумента","Номер строки документа","Номер строки документа","N","4","0","0","1","0"},

{"456","СтрокаИсправления","Номер строки документа-исправления","Строка документа-исправления","N","3","0","0","0","0"}},"15","2","1",

{"FF","202"},

{"Form",

{"2703","ФормаСписка","",""},

{"2707","ПоСотруднику","Открывается из спр. Сотрудники",""}},"2703","2703"},

{"757","Дополнительный","Расчет дополнительной заработной платы","Журнал дополнительной зарплаты","16","2","2450419",

{"CJParams",

{"752","Дни","Дни","Дни","N","6","2","0","0","0"},

{"753","Часы","Часы","Часы","N","7","2","0","0","0"},

{"1236","НомерСтрокиДокумента","Номер строки документа","Номер строки документа","N","4","0","0","0","0"}},"15","2","0",

{"FF","202"},

{"Form",

{"755","ФормаСписка","",""}},"755","755"},

Описание таблицы журнала расчетов "Зарплата" в словаре базы данных представлено фрагментом:

#==TABLE no 123 : Журнал расчетов Зарплата

# Name	Descr	Type[A/S/U]	DBTableName	ReUsable
T=CJ447	Журнал расчетов Зарплата	A	CJ447	1

#-----Fields-------

# Name	Descr	Type	Length	Precision
F=IDDOC		C	9	0
F=IDS		C	9	0
F=IDALG		C	4	0
F=ORDER		N	3	0

F=RESULT	Result	N	15	2
F=DATEB	date1	D	8	0
F=DATEE	date2	D	8	0
F=PERIOD		C	9	0
F=RECALC		N	3	0
F=ID		C	9	0
F=DP		N	1	0
F=IDPARDOC		C	9	0
F=IDRECALC		C	9	0
F=FF202	(P)ОсновнойЭлемент	C	9	0
F=SP448	(P)Дни	N	7	2
F=SP449	(P)Часы	N	8	2
F=SP1089	(P)НомерСтрокиДокуме	N	5	0
F=SP456	(P)СтрокаИсправления	N	4	0

#----Indexes------

# Name	Descr	Unique	Indexed fields	DBName
I=IDDOC+PER		0	IDDOC,PERIOD,IDS,ORDER	IDDOC+PERIO
I=PERIOD+ID		0	PERIOD,IDS,ORDER,DATEB	PERIOD+IDS+
I=IDS+PERIO		0	IDS,PERIOD,ORDER,DATEB	IDS+PERIOD+
I=ID		0	ID	ID
I=IDS+DATEE		0	IDS,DATEE,ID	IDS+DATEE+I
I=DATEE+ID		0	DATEE,ID	DATEE+ID
I=IDPARDOC		0	IDPARDOC	IDPARDOC
I=IDRECALC		0	IDRECALC	IDRECALC
I=FF202	FF202	0	FF202,PERIOD,IDS,ORDER,DATEB	FF202

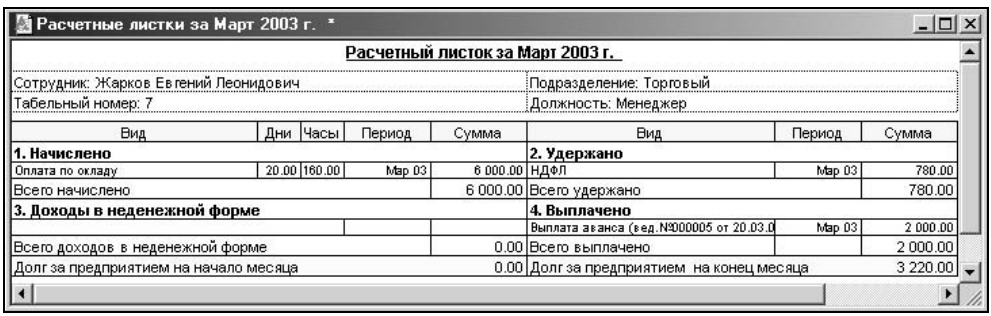
Назначение полей:

- IDDOC — идентификатор документа, которым был задан данный вид расчета;
- IDS — идентификатор сотрудника;
- IDALG — идентификатор вида расчета;
- ORDER — номер вида расчета;
- RESULT — результат расчета;

- DATEB — дата начала периода расчета;
- DATEE — дата окончания периода расчета;
- PERIOD — период (месяц);
- RECALC — вид перерасчета;
- ID — идентификатор строки журнала расчетов;
- DP — признак принадлежности дополнительному журналу;
- IDPARDOC — идентификатор документа, которым была создана данная строка журнала расчета;
- IDRECALC — идентификатор строки журнала-основания для перерасчета;
- FF202 — ОсновнойЭлемент справочника "Сотрудники";
- F=SP448 — количество отработанных дней;
- F=SP449 — количество отработанных часов;
- F=SP1089 — номер строки документа, которым была создана данная строка журнала;
- F=SP456 — строка ручного исправления.

Рассмотрим пример расчета зарплаты (рис. 1.19).

Соответствующий фрагмент таблицы состоит из четырех записей (рис. 1.20).



Расчетные листки за Март 2003 г. *

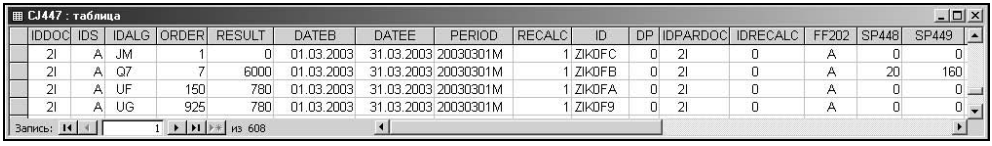
Расчетный листок за Март 2003 г.

Сотрудник: Жарков Евгений Леонидович — Подразделение: Торговый

Табельный номер: 7 — Должность: Менеджер

Вид	Дни	Часы	Период	Сумма	Вид	Период	Сумма
1. Начислено					**2. Удержано**		
Оплата по окладу	20.00	160.00	Мар 03	6 000.00	НДФЛ	Мар 03	780.00
Всего начислено				6 000.00	Всего удержано		780.00
3. Доходы в неденежной форме					**4. Выплачено**		
					Выплата аванса (вед.№000005 от 20.03.0	Мар 03	2 000.00
Всего доходов в неденежной форме				0.00	Всего выплачено		2 000.00
Долг за предприятием на начало месяца				0.00	Долг за предприятием на конец месяца		3 220.00

Рис. 1.19. Расчетный листок сотрудника



CJ447 : таблица

IDDOC	IDS	IDALG	ORDER	RESULT	DATEB	DATEE	PERIOD	RECALC	ID	DP	IDPARDOC	IDRECALC	FF202	SP448	SP449
2I	A	JM	1	0	01.03.2003	31.03.2003	20030301M	1	ZIKDFC	0	2I	0	A	0	0
2I	A	Q7	7	6000	01.03.2003	31.03.2003	20030301M	1	ZIKDFB	0	2I	0	A	20	160
2I	A	UF	150	780	01.03.2003	31.03.2003	20030301M	1	ZIKDFA	0	2I	0	A	0	0
2I	A	UG	925	780	01.03.2003	31.03.2003	20030301M	1	ZIKDF9	0	2I	0	A	0	0

Запись: 1 из 608

Рис. 1.20. Таблица CJ447

Глава 2

Программирование на встроенном языке "1С:Предприятие"

Введение в объектно-ориентированное программирование

В языке программирования, который используется в системе, реализован ряд принципов объектно-ориентированного программирования. Сущность их заключается в следующем:

1. Основными структурными единицами данных являются объекты (метаданные).
2. Каждый объект имеет свойства, которые, в данном случае, определяются его атрибутами.
3. Обращение к объекту допускается через его методы.
4. Методы позволяют изменять свойства объектов или порождать новые (дочерние) объекты.

Используя свойства и методы одного объекта можно порождать дочерние, которые будут наследовать свойства и методы порождающего. Добавление новых свойств и методов, к уже существующим, определяет уникальные отличия порожденного объекта как от родительского, так и от других его дочерних объектов. Обращение к дочернему объекту, свойству или методу осуществляется посредством записи их идентификаторов разделенных точкой. Например, запись — `Таб.ИсходнаяТаблица("СоставСемьи")` означает обращение к свойству `"СоставСемьи"` объекта `Исходная таблица`, который порожден из объекта `Таб`. Запись — `Жильцы.ВыбратьЭлементы()` означает выполнение метода `ВыбратьЭлементы()` над объектом `Жильцы`, который в свою очередь является справочником.

С учетом сказанного, рассмотрим основные конструкции языка программирования и правила их использования.

Программирование на языке 1С

Рассмотрим процесс создания пустой конфигурации со всеми компонентами. Для этого в окне выбора конфигурации, в режиме — Конфигуратор, необходимо выбрать кнопку **Добавить** и в поле **Путь** задать пустую папку (см. рис. 2.1).

После запуска конфигуратора будет создана пустая конфигурация. На примере создания конфигурации "Коммунальные услуги" рассмотрим методы программирования и процессы, связанные с разработкой программных модулей. Для этого создадим основной справочник объектов учета данной конфигурации — "Квартиры" (см. рис. 2.2).

В данном справочнике реквизит **Дом** имеет тип **Справочник.Дома** (см. рис. 2.3).

Справочник "Жильцы" подчинен справочнику "Квартиры" (см. рис. 2.4).

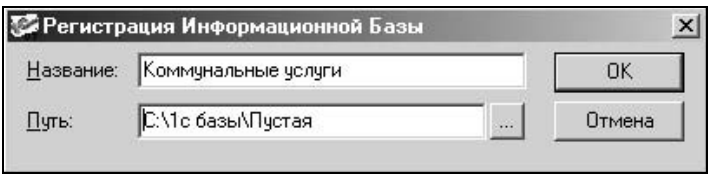


Рис. 2.1. Регистрация пустой конфигурации

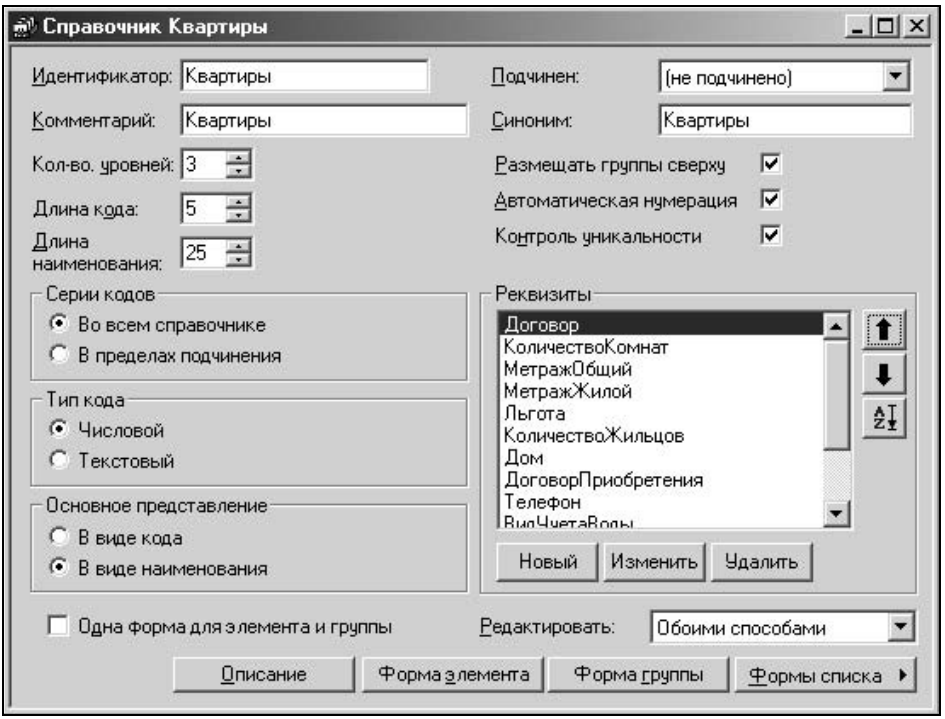


Рис. 2.2. Диалоговое окно **Справочник Квартиры**

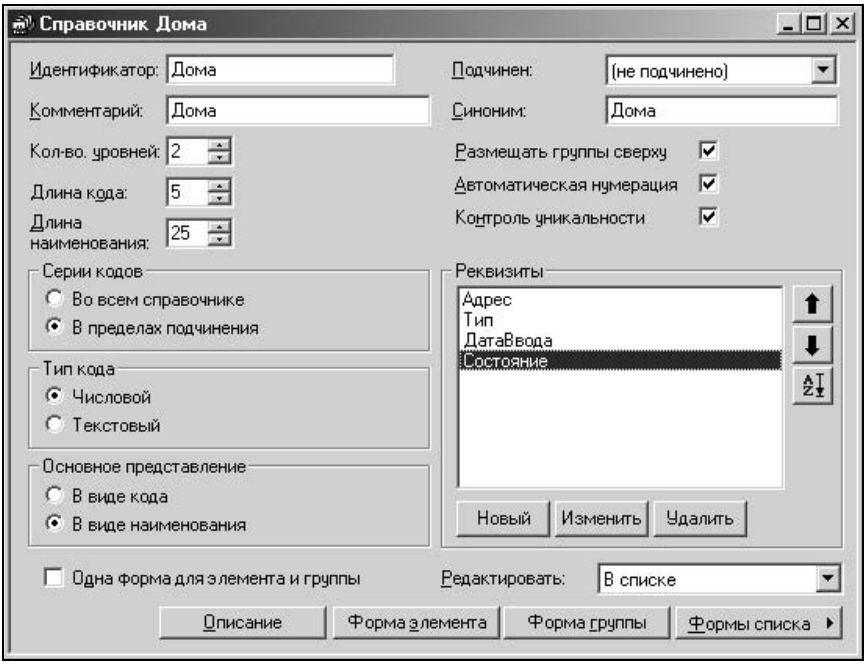


Рис. 2.3. Диалоговое окно **Справочник Дома**

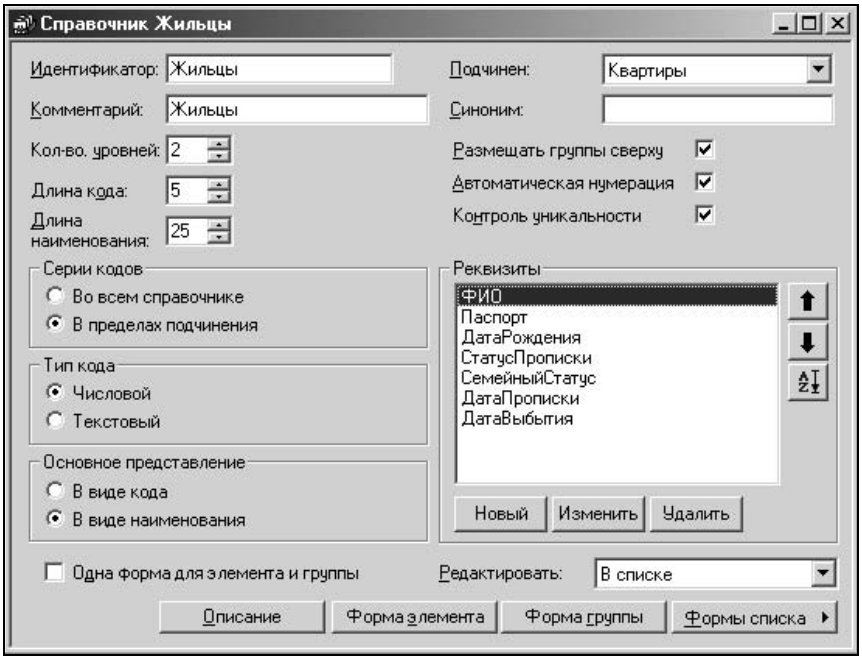


Рис. 2.4. Диалоговое окно **Справочник Жильцы**

Синтаксис и конструкции встроенного языка

В форму списка созданного справочника "Квартиры" (см. рис. 2.5)добавим кнопки **Лицевой счет** с процедурой обработки `ПечатьЛицевогоСчета(1)` и **Состав семьи** с процедурой обработки `ПечатьЛицевогоСчета(0)`.

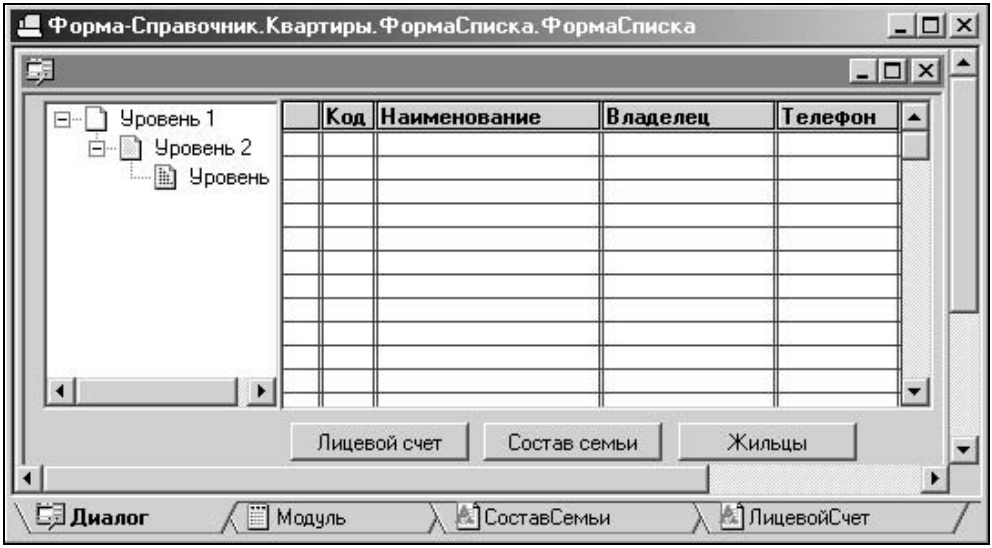


Рис. 2.5. Диалоговое окно формы списка справочника "Квартиры"

Для начала необходимо определить место и правила создания соответствующей процедуры обработки нажатия кнопки. Имя процедуры обработки события, в данном случае — нажатие кнопки, задается на вкладке **Дополнительно** элемента интерфейса в поле **Формула** (см. рис. 2.6).

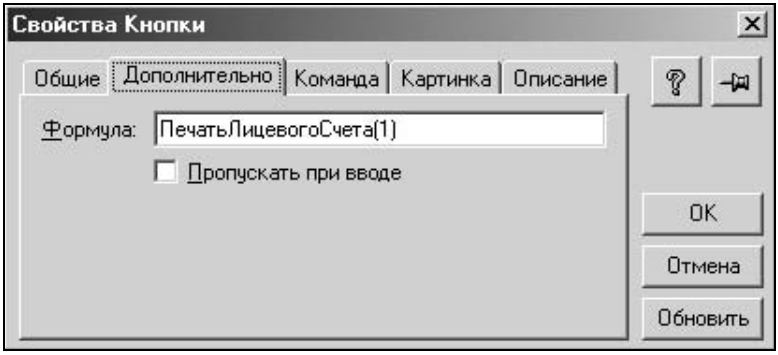


Рис. 2.6. Диалоговое окно свойств элемента интерфейса **Кнопка**

Процедура

На вкладке **Модуль** формы списка справочника создадим процедуру, руководствуясь следующими правилами:

1. Все операторы внутри тела модуля должны заканчиваться символом — , кроме следующих структурных операторов:
   - `Процедура`;
   - `КонецПроцедуры`;
   - `Функция`;
   - `КонецФункции`;
   - `Если…тогда`;
   - `Иначе`;
   - `ИначеЕсли…тогда`;
   - `Пока…цикл`;
   - `Для…цикл`;
   - `Попытка`;
   - `Исключение`.
2. Наличие пробелов между словами в строках не регламентировано.
3. Названия переменных, процедур и функций не должны содержать пробелов, точек, запятых и точек с запятой.
4. Комментарии в тексте выделяются любым количеством символов — /, но не меньше двух.
5. Объявления глобальных переменных должны находится в начале тела модуля до определения первой процедуры или функции.
6. Объявления локальных переменных должны находиться внутри процедуры до первого оператора.
7. Инициализация глобальных переменных должна находиться в конце тела модуля, после определения всех функций и процедур.
8. Первой строкой процедуры должно быть ключевое слово `Процедура`, после которого через любое количество пробелов следует имя процедуры. Ключевое слово `Процедура` начинает секцию исходного текста, выполнение которого можно инициировать из любой точки программного модуля (в данном случае — по нажатию кнопки), просто указав имя процедуры со списком параметров. Если параметры не передаются, то круглые скобки, в объявлении имени, обязательны.

Внимание

Все ключевые слова выделяются в редакторе текста Конфигуратора красным цветом. Если ключевое слово, которое указано курсивом в данном изложении, не выделилось, значит, появилась синтаксическая ошибка.

Примечание

Далее, в тексте программ, все ключевые слова будут выделены жирным цветом.

9. После списка параметров может следовать ключевое слово `Экспорт`, которое указывает на то, что данная процедура является доступной из других программных модулей (это имеет смысл только в глобальном программном модуле).
10. Если в процедуре существует одна или несколько точек возврата в программу, из которой было обращение к процедуре, — используется оператор `Возврат`, который завершает выполнение процедуры и осуществляет этот возврат. Использование данного оператора в процедуре не обязательно.
11. Последняя строка процедуры определяет конец программной секции процедуры ключевым словом `КонецПроцедуры`.

Список параметров

Элементы списка параметров разделяются запятыми. Значения формальных параметров должны соответствовать значениям фактических параметров, передаваемых при вызове процедуры. В этом списке определяются имена каждого из параметров точно так, как они используются в тексте процедуры. Список формальных параметров может быть пуст.

Список параметров может содержать ряд ключевых слов и значений.

- *Ключевое слово* — `Знач`. Это необязательное ключевое слово, которое указывает на то, что следующий за ним параметр передается по значению, т. е. изменение значения формального параметра при выполнении процедуры никак не повлияет на фактический параметр, переданный при вызове процедуры.
- *Имя формального параметра.* После имени формального параметра может следовать знак =, за которым следует значение параметра по умолчанию. Параметры с установленными значениями по умолчанию можно располагать в любом месте списка формальных параметров. Если параметр при вызове процедуры опущен, то он принимает либо установленное по умолчанию значение (если оно есть), либо принимает "пустое" значение (значение неопределенного типа). Если параметру не задано значение по умолчанию и он является последним в списке передаваемых, то при вызове процедуры его нельзя опускать. Если параметру задано значение по умолчанию и он является последним в списке, то при вызове процедуры

его можно опускать в списке передаваемых фактических параметров и не ставить запятую перед опущенным параметром. Если параметру не задано значения по умолчанию, то при вызове процедуры его можно опускать в списке передаваемых фактических параметров, но разделительную запятую надо ставить.

Внимание

Параметр процедуры передается по ссылке, то есть изменение внутри процедуры значения формального параметра приведет к изменению значения соответствующего фактического параметра.

Предварительное объявление процедуры

Если в программном модуле задается вызов процедуры, определение которой находится после определения вызывающей процедуры, то выдается сообщение об ошибке. В тексте программного модуля допускается предварительное описание процедур и функций без их определения. На то, что это предварительное описание, указывает наличие ключевого слова `Далее`, которое замещает в случае предварительного описания тело процедуры и ключевое слово `КонецПроцедуры`. Предварительное описание процедуры или функции может содержаться в любом месте текста модуля, где допускается фактическое определение процедуры или функции, а сам заголовок процедуры (функции) должен в точности соответствовать заголовку в фактическом определении, включая наличие, если необходимо, ключевого слова `Экспорт` и имен формальных параметров.

Процедура — `ПечатьЛицевогоСчета(Режим)`, обспечивающая формирования отчетов "Выписка из лицевого счета" и "Справка о составе семьи", будет выглядеть следующим образом:

```
Процедура ПечатьЛицевогоСчета(Режим)
        Таб=СоздатьОбъект("Таблица");
        Если Режим = 0 тогда
                Таб.ИсходнаяТаблица("СоставСемьи");
        Иначе
                Таб.ИсходнаяТаблица("ЛицевойСчет");
        КонецЕсли;
        Таб.ВывестиСекцию("Шапка");
        Жильцы = СоздатьОбъект("Справочник.Жильцы");
        Жильцы.ИспользоватьВладельца(ТекущийЭлемент());
        Жильцы.ВыбратьЭлементы();
        Пока Жильцы.ПолучитьЭлемент()=1 цикл
                Таб.ВывестиСекцию("Строка");
```

```
    КонецЦикла;
    Таб.Опции(0,0,0,0,"ОпцииСправки");
    Таб.ТолькоПросмотр(1);
    Если Режим = 0 тогда
        Таб.Показать("Справка о составе семьи","");
    Иначе
        Таб.Показать("Выписка из лицевого счета","");
    КонецЕсли;
КонецПроцедуры
```

В данном фрагменте список параметров состоит из одного параметра — `Режим`, значение которого задается в свойствах кнопок.

Функция

Определение функции в программном модуле совпадает с определением процедуры, но с одной существенной разницей. Выполнение функции заканчивается обязательным оператором `Возврат`. Функции отличаются от процедур только тем, что возвращают `ВозвращаемоеЗначение`, представляющее собой выражение, значение которого содержит результат обращения к функции.

Первой строкой функции должно быть ключевое слово `Функция`. Последняя строка функции определяет конец ее программной секции при помощи ключевого слова `КонецФункции`.

Определим функцию формирования колонки "Владелец" в справочнике "Квартиры". Данного реквизита нет непосредственно в справочнике, но в подчиненном справочнике "Жильцы" у элемента есть реквизит — `СтатусПрописки`, который определяет владельца (см. рис. 2.7).

Реквизит `СтатусПрописки` имеет тип `Перечисление.Статусы` (см. рис. 2.8).

Одно из значений перечисления "Статусы" определяет владельца квартиры (см. рис. 2.9).

На вкладке **Дополнительно** свойств текста **Владелец** зададим функцию `ОпределениеВладельца()` (см. рис. 2.10). `ВозвращаемоеЗначение` — должно быть в виде строки текста с "ФИО" владельца квартиры.

Определение данной функции будет выглядеть так:

```
Функция ОпределениеВладельца ()
    Жильцы = СоздатьОбъект("Справочник.Жильцы");
    Жильцы.ИспользоватьВладельца(ТекущийЭлемент());
    Жильцы.ВыбратьЭлементы();
    Пока Жильцы.ПолучитьЭлемент()=1 цикл
Если Жильцы.СтатусПрописки = Перечисление.Статусы.Владелец тогда
```

```
            Возврат Жильцы.ФИО
        КонецЕсли;
    КонецЦикла;
КонецФункции
```

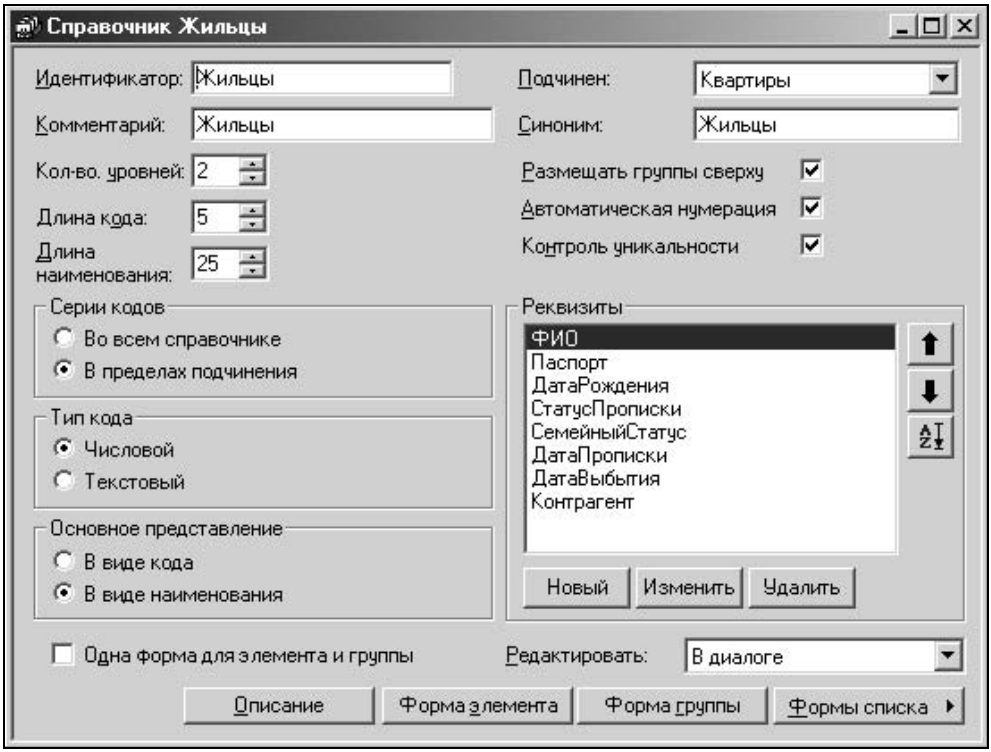


Рис. 2.7. Диалоговое окно справочника "Жильцы"

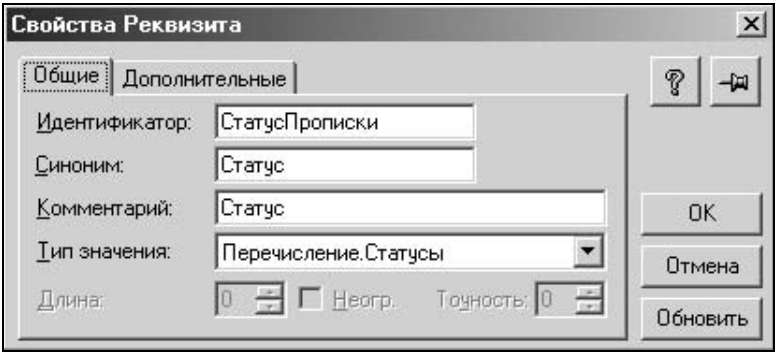


Рис. 2.8. Диалоговое окно свойств реквизита `СтатусПрописки`

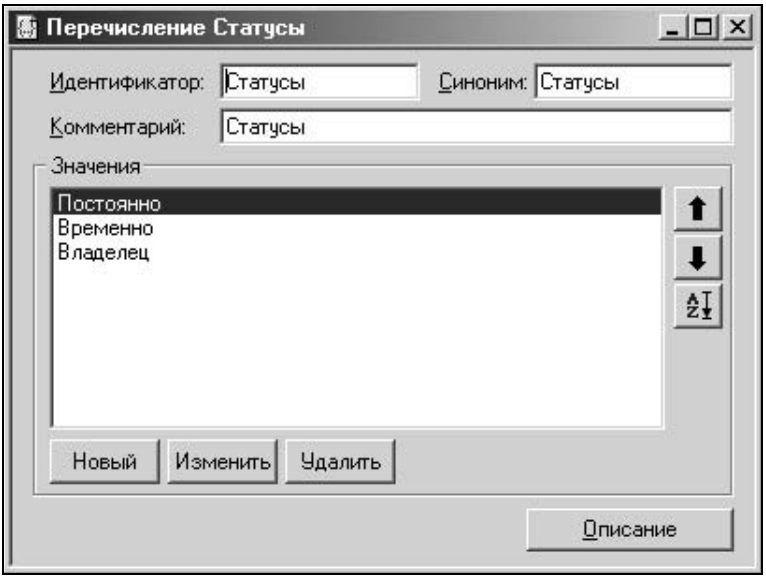


Рис. 2.9. Диалоговое окно свойств реквизита `СтатусПрописки`

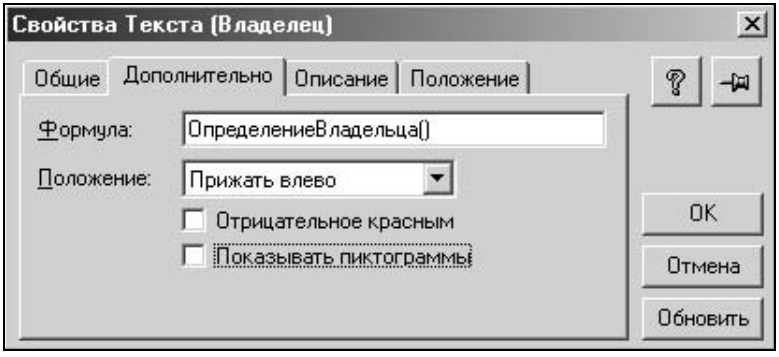


Рис. 2.10. Диалоговое окно свойств текста **Владелец**

Специальные символы

В операторах задания символьных переменных, в случае задания в качестве параметра символьной строки, а также в конструкциях сравнения операторов условного выполнения строки выделяются двойными кавычками.

Для обозначения продолжения символьной строки на следующей строке текста модуля используется символ — |. Например:

```
Предупреждение("Нельзя открыть список жильцов для группы Квартир.
                              |Выберите конкретную Квартиру.");
```

Для комментирования текста модуля используются символы — //, которые вводятся в начале строки комментария. Символов может быть сколько угодно, но не меньше двух. Текст в комментариях кавычками не выделяется.

Оператор условного выполнения

Оператор задания алгоритма выполнения имеет разнообразные формы от самого простого варианта выполнения условия до сложной многовариантной конструкции. Самый простой вариант оператора условного выполнения имеет следующий вид:

```
Если <Логическое_выражение> Тогда
        <операторы>
КонецЕсли;
```

<Логическое выражение> — любая комбинация операндов логического типа в сочетании с логическими операциями И, ИЛИ и НЕ. В тексте функции ОпределениеВладельца() присутствует простой вариант оператора условного выполнения:

```
Если Жильцы.СтатусПрописки = Перечисление.Статусы.Владелец тогда
                Возврат Жильцы.ФИО
        КонецЕсли;
```

Оператор условного выполнения, в котором задается последовательность не только при выполнении определенного условия, но и последовательность операторов в случае невыполнения этого условия, имеет следующий вид:

```
Если <Логическое_выражение> Тогда
        <Последовательность операторов по «истина»>
Иначе
        <Последовательность операторов по «ложь»>
КонецЕсли;
```

Ключевое слово Иначе обозначает начало последовательности операторов в случае невыполнения заданного условия. В приведенном ниже тексте процедуры ПечатьЛицевогоСчета (Режим) присутствует данная конструкция.

```
        Если Режим = 0 тогда
                Таб.ИсходнаяТаблица("СоставСемьи");
        Иначе
                Таб.ИсходнаяТаблица("ЛицевойСчет");
        КонецЕсли;
```

Более сложный оператор условного выполнения с заданием многих последовательностей позволяет вести анализ вариантов условия. Данный оператор является объединением последовательности простых операторов условного выполнения:

```
Если <Логическое_выражение_1> Тогда
```

```
      < Последовательность операторов по –"истина"1>
[ИначеЕсли <Логическое_выражение_к> Тогда
      < Последовательность операторов по – "истина"2>]
[Иначе
      <Последовательность операторов по – "ложь" по всем>]
КонецЕсли;
```

Объединим три кнопки печати формирования отчетов **Лицевой счет**, **Состав семьи** и **Жильцы** в одну кнопку с выбором (см. рис. 2.11).

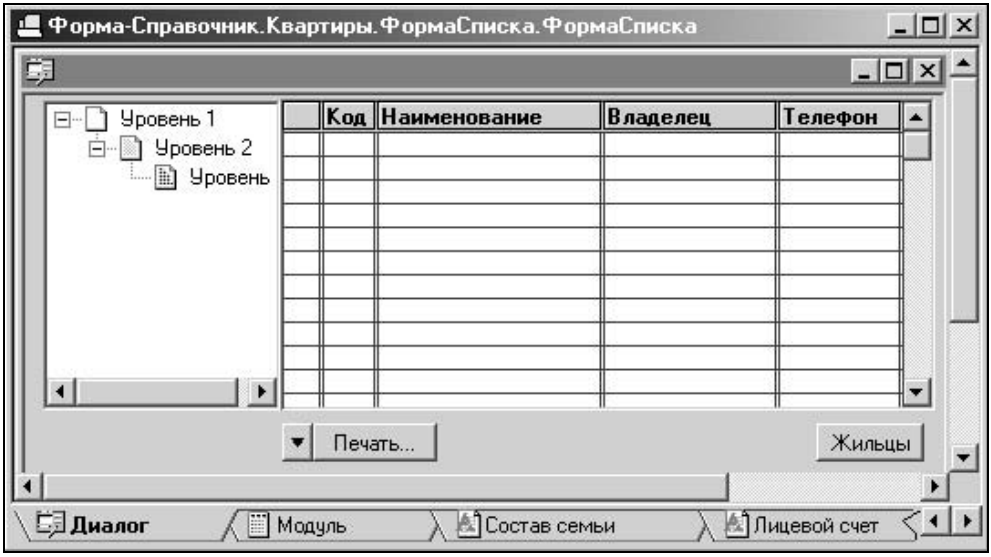


Рис. 2.11. Диалоговое окно формы списка справочника "Квартиры" с кнопкой выбора печатной формы

Кнопка с иконкой ▼ задает выбор из списка печатных форм процедурой `ПоКнопкеВыборПечатнойФормы()` и сохраняет название выбранной формы в названии кнопки **Печать**. Объявим переменную `НазваниеТекущейФормы` глобальной и определим процедуры выполнения по нажатию кнопок ▼ и **Печать**.

```
// Объявление глобальной переменной для модуля формы
Перем НазваниеТекущейФормы;
// Определение процедуры по нажатию на кнопку Печать
Процедура ПоКнопкеПечать()
   Если  ПустоеЗначение(НазваниеТекущейФормы) = 1  Тогда
         НазваниеТекущейФормы = "Состав семьи";
   КонецЕсли;
```

```
   Форма.КнопкаПечать.Заголовок(НазваниеТекущейФормы);
   ПечатьЛицевогоСчета(НазваниеТекущейФормы);
КонецПроцедуры       // ПоКнопкеПечать
//**********************************************************
// Определение процедуры выбора отчета
Процедура ПоКнопкеВыборПечатнойФормы()
СписокПечФорм = СоздатьОбъект("СписокЗначений");
СписокПечФорм.ДобавитьЗначение("Состав семьи");
СписокПечФорм.ДобавитьЗначение("Лицевой счет");
СписокПечФорм.ДобавитьЗначение("Жильцы");
Выбор = СписокПечФорм.ВыбратьЗначение(НазваниеТекущейФормы,,,5, 0);
   Если Выбор = 1 Тогда          // Если выбор сделан
       ПоКнопкеПечать();
   ИначеЕсли Выбор = -1 тогда    // Если истек таймаут
       Сообщить("Выбор формы по умолчанию");
       ПоКнопкеПечать();
   Иначе
Форма.КнопкаПечать.Заголовок("Печать...");    // По кнопке Esc
   КонецЕсли;
КонецПроцедуры          // ПоКнопкеВыборПечатнойФормы
```

В процедуре присутствует рассматриваемая конструкция при анализе действия пользователя:

- был сделан выбор формы;
- истек таймаут;
- нажата кнопка <Esc>.

Подобная конструкция присутствует и в процедуре `ОткрытьЖильцы()`, которая выполняется при нажатии кнопки **Жильцы**.

```
// Определение процедуры вызова подчиненного справочника "Жильцы"
Процедура ОткрытьЖильцы()
   Если ТекущийЭлемент().Выбран() = 0 Тогда
Предупреждение("Нельзя открыть список жильцов, не указав квартиру.
                                        |Введите Квартиру.");
   ИначеЕсли ТекущийЭлемент().ЭтоГруппа() = 1 Тогда
Предупреждение("Нельзя открыть список жильцов для группы Квартир.
                          |Выберите конкретную Квартиру.");
   Иначе
              ОткрытьФорму("Справочник.Жильцы");
   КонецЕсли;
КонецПроцедуры // ОткрытьЖильцы
```

В данном случае ведется анализ положения указателя в справочнике "Квартиры":

- справочник пустой;
- указатель установлен на группе;
- указатель установлен на конкретной квартире.

Оператор цикла

Многократно выполняемая последовательность операторов объединяется в конструкцию цикла. Цикл с предусловием выполняется до тех пор, пока выполняется заданное условие. Например:

```
Пока  <Логическое_выражение> Цикл
        <Последовательность операторов>
КонецЦикла;
```

<Логическое выражение> задает условие выполнения.

В процедуре `ПечатьЛицевогоСчета(Имя)` присутствует данная конструкция:

```
Пока Жильцы.ПолучитьЭлемент()=1 цикл
   Таб.ВывестиСекцию("Строка");
КонецЦикла;
```

В приведенном фрагменте из подчиненного справочника "Жильцы" выбираются элементы и выводятся в отчет до тех пор, пока функция `ПолучитьЭлемент()` не вернет 0, что означает, что достигнут конец справочника.

Цикл с шагом повторяется заданное счетчиком количество раз:

```
Для  <Имя_переменной> = <Выражение1> По <Выражение2> Цикл
       <Последовательность операторов>
КонецЦикла;
```

Здесь:

- `<Имя_переменной>` — идентификатор переменной (счетчик цикла);
- `<Выражение1>` — начальное значение счетчика цикла;
- `<Выражение2>` — конечное значение счетчика цикла;
- `<операторы>` — последовательность исполняемых операторов.

Обычно оператор используется при работе со списками значений и с таблицами. В этих объектах метаданных, как правило, на определенный момент можно получить количество строк в списке или таблице.

Оператор *Попытка...Исключение*

В процессе выполнения программы возникают ситуации, которые невозможно обработать обычным анализом с помощью операторов условного выполнения. Для управления такими ошибочными (исключительными) ситуа-

циями предназначен оператор Попытка…Исключение. В качестве ошибочных (исключительных) ситуаций рассматриваются ошибки времени выполнения модуля. Для диагностики ошибки обычно используется системная функция ОписаниеОшибки(), которая возвращает описание ошибки времени выполнения модуля, такое же, как то, которое выдается в окне сообщений.

Синтаксис оператора:

```
Попытка
        <Последовательность операторов>
Исключение
        <Обработка исключительной ситуации>
КонецПопытки;
```

Функция ОписаниеОшибки() предназначена для использования в группе <Обработка исключительной ситуации>.

Самый простой пример использования данной конструкции является обработка возможной ошибки выполнения при загрузке данных из файла, например, из конфигурации "Зарплата и кадры":

```
Попытка
                Правила.Загрузить(ФормИмяФайлаПравил);
Исключение
                Предупреждение(ОписаниеОшибки());
                Возврат(0);
КонецПопытки;
```

В качестве более сложного примера с использованием функций НачатьТранзакцию() и ЗафиксироватьТранзакцию() рассмотрим фрагмент из обработки ЗаписьПериодическихРеквизитов(). Эта обработка вызывается в предопределенной процедуре форм элементов справочников, в которых есть периодические реквизиты, ПриЗаписи(). В нашем справочнике "Квартиры" реквизит "Льгота" периодический. Поэтому вставим в предопределенную процедуру ПриЗаписи() вызов общей функции для всех справочников глБухЗаписьПериодическихРеквизитов(), которая находится в *Глобальном модуле*. В качестве параметра передадим контекст формы:

```
// Предопределенная процедура.
    Процедура ПриЗаписи()
     СтатусВозврата(глБухЗаписьПериодическихРеквизитов(Контекст));
    КонецПроцедуры          // ПриЗаписи()
```

Запись периодических реквизитов требует комбинации действий по удалению и записи сразу нескольких записей в базе данных. Во время выполнения этих действий может возникнуть исключительная ситуация.

```
    Процедура ЗаписатьЗначения()
    Периодический = СоздатьОбъект("Периодический");
```

```
    Попытка
        НачатьТранзакцию();
// При первой записи элемента системой принудительно записываются
первые значения периодических реквизитов — их надо удалить.
ПерваяЗапись=ПустоеЗначение(КонтекстФормы.ТекущийЭлемент());
КонтекстФормы.Записать();
ТекущийЭлемент = КонтекстФормы.ТекущийЭлемент();
Если ПерваяЗапись = 1 Тогда
Для Номер = 1
По Метаданные.Справочник(КонтекстФормы.Вид()).Реквизит() Цикл
мдРекви-
зит = Метаданные.Справочник(КонтекстФормы.Вид()).Реквизит(Номер);
Если мдРеквизит.Периодический = 0 Тогда
      Продолжить;
КонецЕсли;
Если (ТекущийЭлемент.ЭтоГруппа() = 0) И (мдРеквизит.Использование =
"ДляГруппы") Тогда
      Продолжить;
КонецЕсли;
Если (ТекущийЭлемент.Этогруппа() = 1) И (мдРеквизит.Использование =
"ДляЭлемента") Тогда
      Продолжить;
КонецЕсли;
Периодический.ИспользоватьОбъект(мдРеквизит.Идентификатор, ТекущийЭ-
лемент);
Периодический.ВыбратьЗначения();
Пока Периодический.ПолучитьЗначение() = 1 Цикл
      Периодический.Удалить();
КонецЦикла;
КонецЦикла;
КонецЕсли;
Значения.ВыбратьСтроки();
Пока Значения.ПолучитьСтроку() = 1 Цикл
      Если Значения.Пометка <> 2 Тогда // пометка выключена
            Продолжить;
      КонецЕсли;
      Если Периодический.ИспользоватьОбъект(Значения.Идентификатор,
      ТекущийЭлемент) = 1 Тогда
            ТипВид = ТипЗначенияСтр(Значения.НовоеЗначение);
            Длина = 0;
```

```
                Точность = 0;
                Если       (ТипВид = "Число") Тогда
                        Длина     = СтрДлина(Значения.НовоеЗначение);
Точность=Длина-Мин(Длина,Найти(Строка(Значения.НовоеЗначение)
                                                +".", "."));
                ИначеЕсли (ТипВид = "Строка") Тогда
                        Длина = СтрДлина(Значения.НовоеЗначение);
                ИначеЕсли (ТипВид = "Справочник") ИЛИ (ТипВид = "Документ") ИЛИ (ТипВид = "Перечисление") ИЛИ (ТипВид = "Счет") ИЛИ (ТипВид = "Календарь") Тогда
                ТипВид    = ТипВид+"."+Значения.НовоеЗначение.Вид();
                КонецЕсли;
                Периодический.НазначитьТип(ТипВид, Длина, Точность);
                Периодический.Значение = Значения.НовоеЗначение;
                Если (Значения.СтараяДатаЗнач = '00.00.0000') И (ПерваяДата <>   '00.00.0000') Тогда
// Первые значения периодических реквизитов в некоторых  случаях следует записывать на заведомо давнюю дату.
                        Периодический.ДатаЗнач = ПерваяДата;
                Иначе
                        Периодический.ДатаЗнач = НоваяДата;
                КонецЕсли;
                Периодический.Записать();
        КонецЕсли;
        КонецЦикла;
        ЗафиксироватьТранзакцию();
        // Признак успешной записи.
        Форма.Параметр = 1;
        Форма.Закрыть();
Исключение
        Сообщить(ОписаниеОшибки(), "!");
КонецПопытки;
КонецПроцедуры  // ЗаписатьЗначения()
```

Как видно из фрагмента, в случае возникновения исключительной операции транзакция не фиксируется.

Оператор *ВызватьИсключение*

Данный оператор применяется в языковой конструкции `ПопыткаИсключение … КонецПопытки`. Обработка исключительных ситуаций могут быть вложенной. При этом, при возникновении исключительной ситуации, управление будет

передано на самый последний, по вложению, обработчик, начинающийся с ключевого слова Исключение. Последовательность операторов обработки исключительной ситуации может содержать оператор ВызватьИсключение. Выполнение данного оператора прекращает выполнение последовательности обработки исключительной ситуации. При этом производится поиск "внешнего" обработчика. Если таковой есть, то управление передается на его первый оператор. Если нет, то выполнение модуля прекращается с выдачей сообщения о первоначально возникшей ошибке. Таким образом, организуется цикл ожидания "благоприятного" для обработчика события. Оператор ВызватьИсключение может встречаться только внутри операторных скобок Исключение ... КонецПопытки.

Оператор *Перейти*

Оператор безусловного перехода осуществляет передачу управления на отмеченный оператор в теле модуля.

Синтаксис:

```
Перейти <Метка>;
```

Метка ставится перед тем оператором, к которому производится переход, после метки следует символ ":". Для обозначения объекта типа «метка» используется символ «~», который вводится пред меткой.

Использование оператора безусловной передачи управления на исполняемый оператор программного блока считается признаком "плохого тона" в объектно-ориентированном программировании. Однако оператор, в определенных случаях, все еще используется в программировании. Например, в отчете ПодготовкаСведенийДляПФР:

```
Процедура ЗаписатьИПоказатьТекст(ВыходнойТекст,ИмяФайла,
                                                ЗаголовокОкна)
    ФайлЗаписан = 1;
    Если Найти("А,В",Лев(ИмяФайла,1)) > 0 Тогда
        ПутьВывода = Лев(ИмяФайла,2);
        ТекстВопроса = "Поставьте чистую дискету для файла пачки
        "+Сред(ЗаголовокОкна,Найти(ЗаголовокОкна,"№"));
        ~Начало:Хорошо=0;
        Если Вопрос(ТекстВопроса,"ОК+Отмена")="ОК"  Тогда
            Если ФС.СуществуетФайл(ПутьВывода+"\NUL")=0  Тогда
                ТекстВопроса = "Поставьте чистую дискету для файла пачки
                "+Сред(ЗаголовокОкна,Найти(ЗаголовокОкна,"№"));
                Перейти ~Начало;
            КонецЕсли;
            Если ФС.СуществуетФайл(ИмяФайла)=1  Тогда
                ФС.УдалитьФайл(ИмяФайла)
```

```
        КонецЕсли;

        Если ПустаяСтрока(ФС.НайтиПервыйФайл(ПутьВывода + "\" +
            Прав(СокрЛП(глПолучитьНалог("ПФР_страх").РегНомер),6)
            +"??.*")) = 0  Тогда

            ТекстВопроса  = "На дискете присутствует файл, относя-
            щийся к другой пачке!"+РазделительСтрок+"Поставьте чис-
            тую дискету для файла пачки
            "+Сред(ЗаголовокОкна,Найти(ЗаголовокОкна,"№"));

                  Перейти ~Начало;

            КонецЕсли;

            Иначе

                  ФайлЗаписан = 0;

            КонецЕсли;

      КонецЕсли;

      Если ФайлЗаписан = 1 Тогда

            ВыходнойТекст.Записать(ИмяФайла);

            Если ФС.СуществуетФайл(ИмяФайла) = 0 Тогда

                  Предупреждение("Не возможен вывод файла данных
                  для передачи в ПФР в каталог "+
                  Лев(ИмяФайла,Найти(ИмяФайла,"\")-1)+"!");

                  ФайлЗаписан = 0;

            КонецЕсли;

      КонецЕсли;

      Если ФайлЗаписан = 1 Тогда

            ТекстДляПоказа = СоздатьОбъект("Текст");

            ТекстДляПоказа.КодоваяСтраница(1);

            ТекстДляПоказа.Открыть(ИмяФайла);

            ТекстДляПоказа.Показать(ЗаголовокОкна);

      Иначе

            Выходной-
Текст.Показать(ЗаголовокОкна,Прав(ИмяФайла,12));
      КонецЕсли;

      КонецПроцедуры       // ЗаписатьИПоказатьТекст()
```

Оператор *Продолжить*

Оператор передачи управления в начало цикла используется, когда необходимо прервать выполнение шага цикла и перейти к началу следующего шага. Рассмотрим процедуру формирования лицевого счета `ПечатьЛицевогоСчета(Имя)`. Добавим анализ помеченных на удаление элементов:

```
    Пока Жильцы.ПолучитьЭлемент()=1 цикл

      Если Жильцы.ПометкаУдаления()=1 тогда

            Продолжить;
```

```
    КонецЕсли;
        Таб.ВывестиСекцию("Строка");
    КонецЦикла;
```

Если элемент справочника помечен на удаление, строка данного элемента в отчет выведена не будет.

Оператор *Прервать*

Это оператор принудительного завершения выполнения цикла. Используется, когда необходимо прервать выполнение всего цикла. В предопределенной процедуре `ОбработкаПроведения()` модуля документа `ПоказанияВодомеров`, которая будет рассмотрена в разделе "Предопределенные процедуры модуля документа", присутствует данная конструкция:

```
Пока Жильцы.ПолучитьЭлемент()=1 цикл
  Если Жильцы.СтатусПрописки=Перечисление.Статусы.Владелец тогда
// Владелец найден, дальнейшее выполнение цикла необходимо прервать
      ТаблицаНачислений.НоваяСтрока();
      ТаблицаНачислений.Контрагент = Жильцы.Контрагент;
      ТаблицаНачислений.Сумма = ЖурналРасчетовНач.Результат;
      Прервать;
  КонецЕсли;
КонецЦикла;
```

Оператор *Возврат*

Оператор завершения процедуры или функции используется в случае, когда по данной ветке процедуры или функции достигнут логический конец. По данному оператору происходит возврат в точку вызова процедуры или функции.

Синтаксис:

```
Возврат [<Выражение>]
```

<Выражение>, после ключевого слова `Возврат`, представляет собой значение, возвращаемое функцией.

Внимание

<Выражение> имеет смысл только для функции.

Рассмотрим функцию `ОпределениеВладельца()`, которая была рассмотрена в разделе "Функция". Оператором

```
    Возврат Жильцы.ФИО
```

в графу "Владелец" возвращается текущий элемент справочника "Жильцы".

Директивы

Директива переключения загрузки программного модуля на загрузку из текстового файла — ЗагрузитьИзФайла, используется в тех случаях, когда обновление конфигурации непосредственно через Конфигуратор затруднительно. В таких случаях достаточно изменить содержимое текстового файла.

Синтаксис:

#ЗагрузитьИзФайла <ИмяФайла>

Параметр — <ИмяФайла>, определяет имя файла, содержащего исходный текст программного модуля (записывается без кавычек и скобок).

Внимание

Конструкция **#ЗагрузитьИзФайла** <ИмяФайла> должна записываться в первой строке программного модуля с первой позиции.

Создадим текстовый файл в корневом каталоге диска С: с именем — Загрузка.txt. Скопируем содержимое тела модуля формы списка справочника "Квартиры":

```
// Определение процедуры по нажатию на кнопку Печать
Процедура ПоКнопкеПечать()
        Если  ПустоеЗначение(НазваниеТекущейФормы) = 1  Тогда
                НазваниеТекущейФормы = "Состав семьи";
        КонецЕсли;
        Форма.КнопкаПечать.Заголовок(НазваниеТекущейФормы);
        ПечатьЛицевогоСчета(НазваниеТекущейФормы);
КонецПроцедуры          //ПоКнопкеПечать
//*****************************************************************
// Определение процедуры выбора отчета
Процедура ПоКнопкеВыборПечатнойФормы()
        СписокПечФорм = СоздатьОбъект("СписокЗначений");
        СписокПечФорм.ДобавитьЗначение("Состав семьи");
        СписокПечФорм.ДобавитьЗначение("Лицевой счет");
        СписокПечФорм.ДобавитьЗначение("Жильцы");
        Выбор = СписокПечФорм.ВыбратьЗначение
                        (НазваниеТекущейФормы,,,5, 0);
        Если Выбор = 1 тогда                    // Если выбор сделан
                ПоКнопкеПечать();
        ИначеЕсли Выбор = -1 тогда
        Сообщить("Выбор формы по умолчанию");  // Если истек таймаут
                ПоКнопкеПечать();
```

```
        Иначе
        Форма.КнопкаПечать.Заголовок("Печать..."); // По кнопке Esc
        КонецЕсли;
КонецПроцедуры //ПоКнопкеВыборПечатнойФормы
//*********************************************************
// Определение прцедуры формирования отчета
Процедура   ПечатьЛицевогоСчета  (Имя)
        Таб=СоздатьОбъект("Таблица");
        Таб.ИсходнаяТаблица(Имя);
        Таб.ВывестиСекцию("Шапка");
        Жильцы = СоздатьОбъект("Справочник.Жильцы");
        Жильцы.ИспользоватьВладельца(ТекущийЭлемент());
        Жильцы.ВыбратьЭлементы();
        Пока Жильцы.ПолучитьЭлемент()=1 цикл
                Если Жильцы.ПометкаУдаления() = 1 тогда
                        Продолжить;
                КонецЕсли;
                Таб.ВывестиСекцию("Строка");
        КонецЦикла;
        Таб.Опции(0,0,0,0,"ОпцииСправки");
        Таб.ТолькоПросмотр(1);
        Таб.Показать(Имя,"");
КонецПроцедуры  //ПечатьЛицевогоСчета
//*****************************************************
// Определение процедуры вызова окна подчиненного справочника "Жильцы"
Процедура ОткрытьЖильцы()
        Если ТекущийЭлемент().Выбран() = 0 Тогда
         Предупреждение("Нельзя открыть список жильцов,
                        |не указав квартиру.Введите Квартиру.");
        ИначеЕсли ТекущийЭлемент().ЭтоГруппа() = 1 Тогда
            Предупреждение("Нельзя открыть список жильцов
                  |для группы Квартир.Выберите конкретную Квартиру.");
        Иначе
                ОткрытьФорму("Справочник.Жильцы");
        КонецЕсли;
КонецПроцедуры  //ОткрытьЖильцы
```

В модуле формы списка справочника "Квартиры" зададим директиву загрузки текста из файла:

```
#ЗагрузитьИзФайла C:\Загрузка.txt
```

В случае отсутствия заданного файла пользователю будет выдано сообщение:

Не могу открыть файл: <имя файла>

Алгоритм исполнения модулей встроенного языка

В Конфигураторе, в диалогах задания свойств большинства объектов метаданных, присутствуют вкладки **Модуль** и **Модуль Документа** (для документов). В каком же порядке исполняется набор процедур в модулях?

При нажатии пользователем кнопки, или при вводе данных в поле ввода, а также при задании признака — исполняется процедура или функция, определенные в модуле формы. Кроме выполнения процедур и функций, назначенных для исполнения элементам интерфейса, существуют и так называемые "предопределенные" процедуры. Процедуры, назначенные для исполнения объектам метаданных для обработки интерактивных действий пользователя, называются предопределенными процедурами.

Процедуры и функции элементов формы

Процедуры и функции, исполняемые при действиях пользователя с элементами формы, задаются в свойствах элемента на вкладке **Дополнительно** в поле **Формула**. Имя процедуры или функции должно совпадать с именем которое определено для нее в модуле формы процедуры. В предыдущем разделе был создан модуль формы списка справочника "Квартиры". Процедура `ПоКнопкеВыборПечатнойФормы()` была назначена на элемент интерфейса типа **Кнопка** с иконкой  (см. рис. 2.12).

Следует обратить внимание на то, что на вкладке **Дополнительно** в качестве формулы чаще всего задаются процедуры или функции.

Характерной процедурой, отрабатывающей действия при вводе в несколько полей, является функция пересчета в документах. Создадим документ ввода показаний водомеров (см. рис. 2.13).

Создадим справочник "Услуги" (см. рис. 2.14). Реквизиту шапки **Услуга** документа "ПоказанияВодомеров" зададим тип **Справочник.Услуги**.

Реквизит справочника "Услуги" — **Цена.** Этот реквизит периодический для ведения истории изменения тарифов на услуги.

Назначим в форме документа **Показания водомеров** реквизитам **Квартира**, **ПоказаниеСчетчика** (**Показание [куб. м3]**) и **Цена** процедуру `Пересчет()` (см. рис. 2.15).

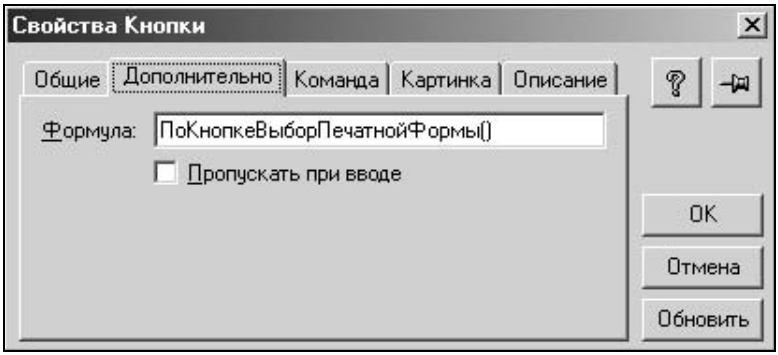


Рис. 2.12. Диалоговое окно свойств кнопки на вкладке **Дополнительно**

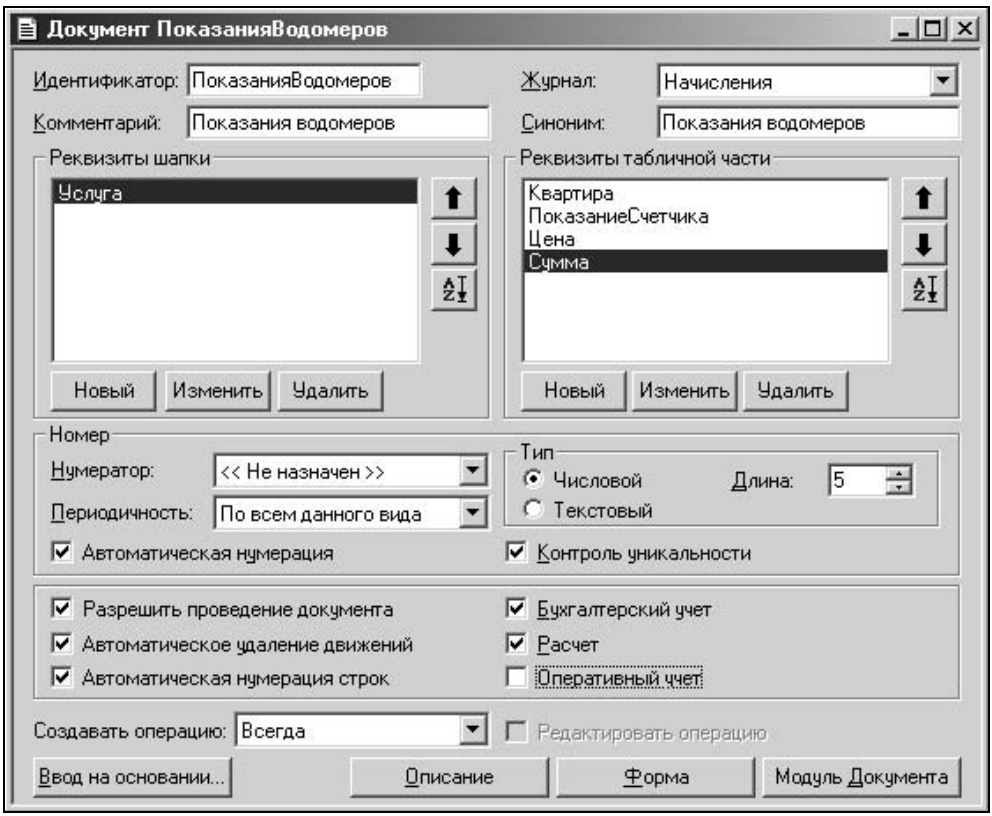


Рис. 2.13. Диалоговое окно задания документа **ПоказанияВодомеров**

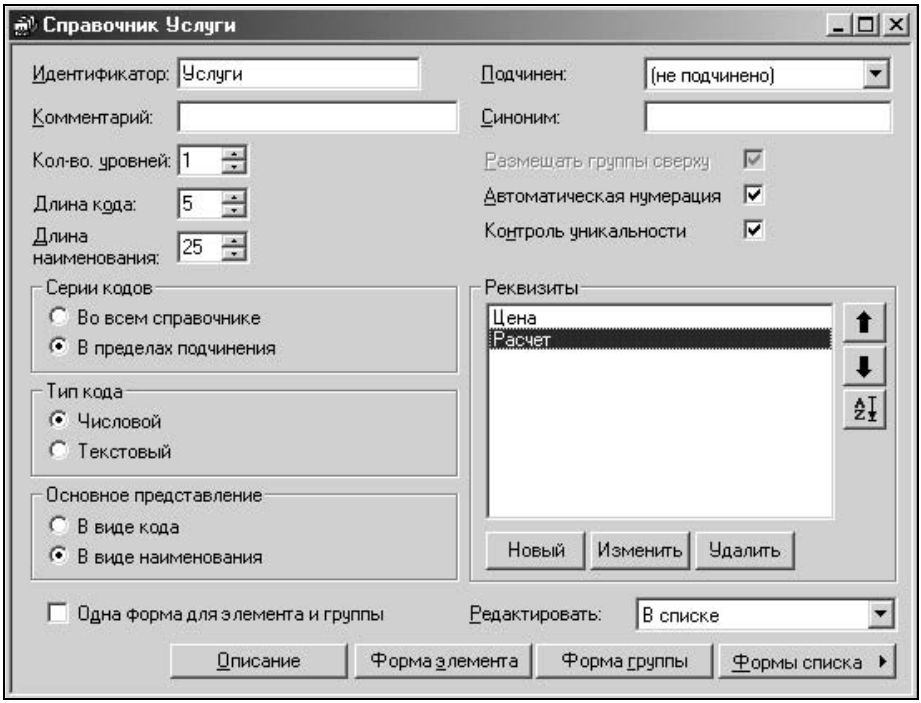


Рис. 2.14. Диалоговое окно **Справочник Услуги**

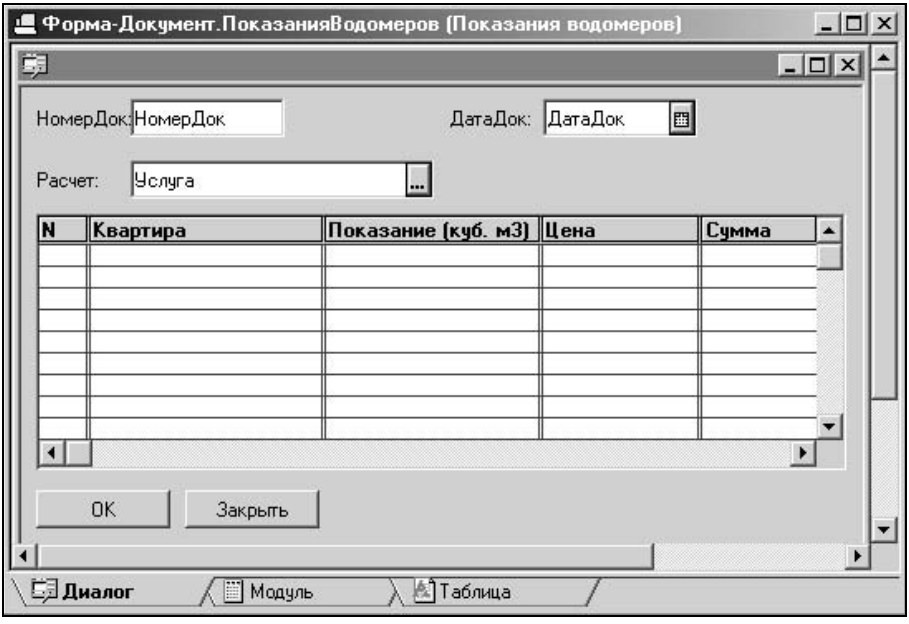


Рис. 2.15. Диалоговое окно формы документа **ПоказанияВодомера**

При выборе квартиры из справочника, вводе показания счетчика или изменения цены, будет выполняться процедура, которая производит пересчет всех реквизитов табличной части.

```
Процедура Пересчет()
    Цена = Услуга.Цена.Получить(ДатаДок);
    // Получим предыдущее показание счетчика
    ПоказаниеПр= Квартира.ПоказаниеХолВоды.Получить (НачМеся-
    ца(ДатаДок)));
    Если Квартира.Льгота.Получить(ДатаДок)=1 тогда
        Цена = Цена*Льгота.Получить(ДатаДок);
    КонецЕсли;
    Сумма = Цена*(ПоказаниеСчетчика - ПоказаниеПр);
КонецПроцедуры
```

Предопределенные процедуры

В модуле формы можно определить процедуры, которые выполняются при задании пользователем определенных действий, таких как открытие документа, элемента справочника, создании нового документа или элемента справочника, записи документа или справочника, при вводе документа на основании другого и т.п. Предопределенные процедуры имеют специальные имена и набор параметров, причем они, для модуля формы и формы списка, отличаются.

Статус возврата

Во всех предопределенных процедурах имеется возможность отменить выполнение действия, для обработки которого процедура предназначена. Для этого предназначена специальная функция `СтатусВозврата(Статус)`.

Если необходимо отменить действие, которое обрабатывает предопределенная процедура, то значение параметра `Статус` — 0, в противном случае значение параметра `Статус` будет 1.

Начальное значение статуса возврата предопределенной процедуры равное 1 (выполнить действие), устанавливается системой при вызове предопределенной процедуры.

Предопределенные процедуры справочников

Форма элемента справочника может быть только одна. Форм списка справочника может быть сколько угодно. Предопределенные процедуры задаются для формы элемента справочника в модуле формы элемента справочника. Для форм списка предопределенные процедуры задаются в модулях форм списка.

Предопределенные процедуры формы элемента справочника

Рассмотрим предопределенные процедуры для формы элемента справочника.

1. `Ввод нового()`.

Процедура `ВводНового(ПризнакКопирования,ОбъектКопирования)` выполняется при интерактивном вводе нового элемента справочника. Система 1С:Предприятие передает процедуре два параметра:

- `ПризнакКопирования` имеет значение 1, если новый элемент справочника создается путем копирования;
- `ОбъектКопирования` — если параметр `ПризнакКопирования` имеет значение 1, в таком случае в распоряжении программиста оказывается `ОбъектКопирования`.

Как правило, данная процедура применяется для задания реквизитам значений "по умолчанию". Например, в нашем справочнике "Квартиры" можно задать в модуле формы элемента количество жильцов по умолчанию — `1` и льгота `1` следующим образом:

```
Процедура ВводНового(Копирование)
  Если Копирование = 0 тогда
     КоличествоЖильцов = 1;
     Льгота = 1;
  КонецЕсли;
КонецПроцедуры // ВврдНового()
```

2. `ПриЗаписи()`.

Процедура `ПриЗаписи(СписокПериодическихРеквизитов)` выполняется при интерактивной записи элемента справочника.

Параметром процедуры является — `СписокПериодРекв`. Он представляет строку со списком изменяемых периодических реквизитов справочника. В данный параметр система 1С:Предприятие передает перечень периодических реквизитов, которые были интерактивно выбраны пользователем для обновления в диалоговом окне выбора. В теле процедуры значение данного параметра может быть изменено, что позволяет в данной процедуре непосредственно управлять списком записываемых значений периодических реквизитов. В данной главе, в разд. "Попытка исключение", был приведен фрагмент из обработки `ЗаписьПериодическихРеквизитов`, в которой пользователю предлагается диалоговое окно для задания даты периодических реквизитов с информацией о каждом из них.

Вызов обработки задается из процедуры `ПриЗаписи()`:

```
Процедура ПриЗаписи()
  СтатусВозврата(глБухЗаписьПериодическихРеквизитов(Контекст));
КонецПроцедуры          // ПриЗаписи()
```

3. ПриОткрытии().

Процедура ПриОткрытии() выполняется при интерактивном открытии формы элемента справочника.

Внимание

В вызове помощи, следует иметь в виду, что синтакс процедур ПриОткрытии() и ПриЗакрытии() отнесены в раздел предопределенных процедур форм списка справочника. На самом деле эти процедуры выполняются при интерактивном открытии и закрытии как формы элемента справочника, так и формы списка.

Определим процедуру ПриОткрытии() в модуле формы элемента справочника "Квартиры" для управления количеством закладок в форме. Введем новый реквизит в справочник **ВидУчетаВоды** (см. рис. 2.16) и реквизиты показаний водомеров **ПоказаниеГорВоды** и **ПоказаниеХолВоды**.

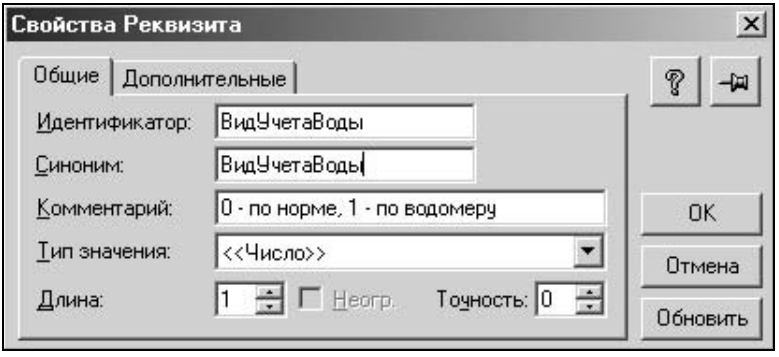


Рис. 2.16. Диалоговое окно свойств реквизита **ВидУчетаВоды**

Создадим два новых слоя для распределения реквизитов с учетом вида учета воды.

Слой "Основной" содержит общие реквизиты (см. рис. 2.17).

Слой "Кнопки" содержит кнопки и признак учета воды по водомеру (см. рис. 2.18).

Слой "ПоказанияСчетчиков" содержит реквизиты ввода показаний и кнопку **История** (см. рис. 2.19).

В предопределенной процедуре ПриОткрытии() ведется анализ реквизита **ВидУчета воды** для активизации новой закладки:

```
Процедура ПриОткрытии()
  Если ВидУчетаВоды = 1 тогда
     Форма.ИспользоватьЗакладки(1);
     Форма.Закладки.ДобавитьЗначение("Общий","Общие");
```

```
        Форма.Закладки.ДобавитьЗначение("ПоказанияСчетчиков",
                                        "Показания счетчиков");
    КонецЕсли;
    Форма.ИспользоватьСлой("Основной, Кнопки", 2);
КонецПроцедуры // ПриОткрытии
```

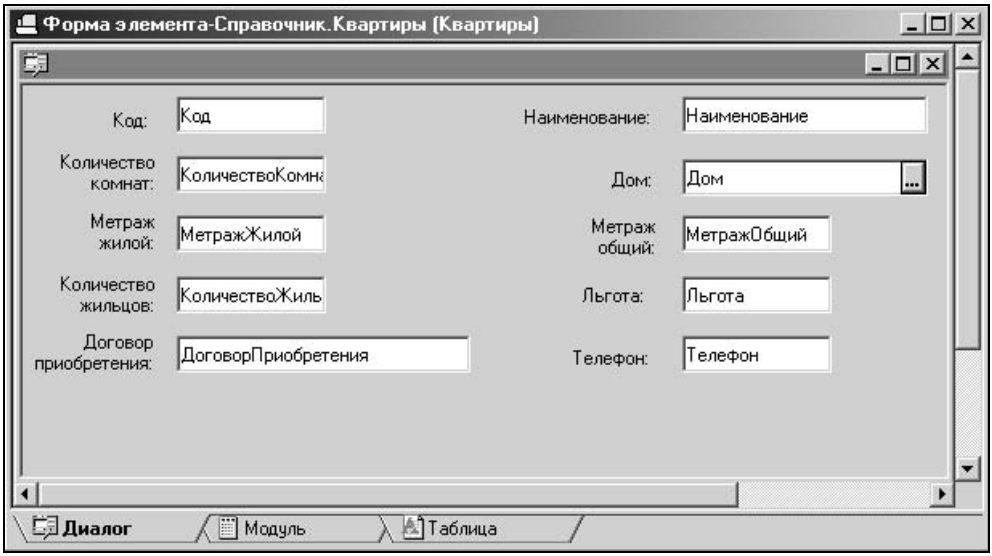


Рис. 2.17. Диалоговое окно справочника "Квартир", слой "Основной"

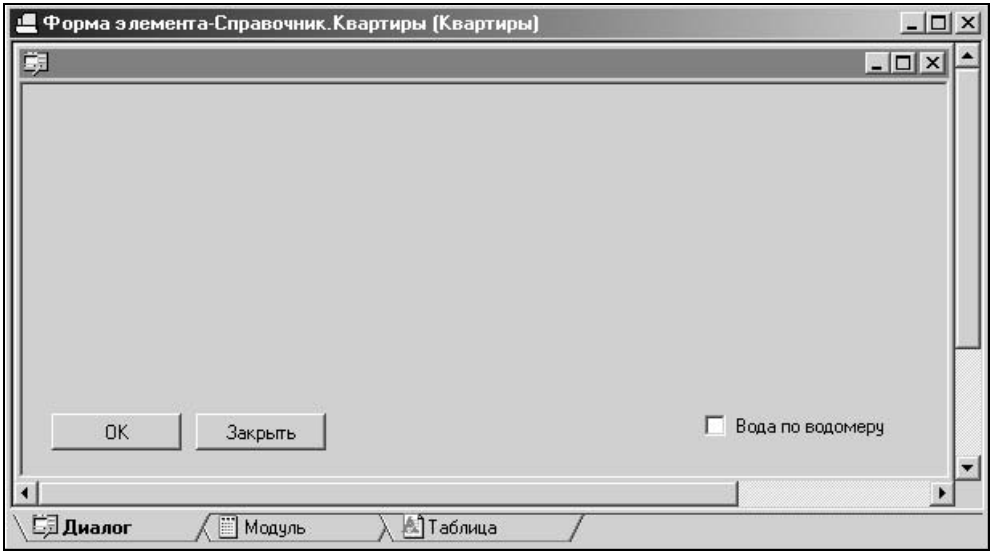


Рис. 2.18. Диалоговое окно справочника "Квартиры", слой "Кнопки"

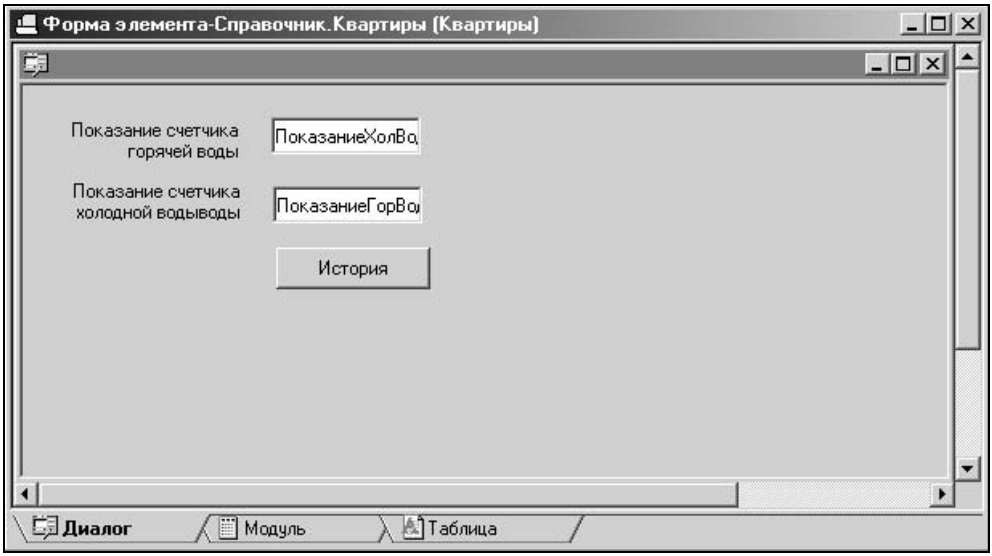


Рис. 2.19. Диалоговое окно справочника "Квартиры", слой "ПоказанияСчетчиков"

Целесообразно рассмотреть предопределенную процедуру контекста формы `ПриВыбореЗакладки`, которую мы тоже определим в модуле формы элемента справочника:

```
Процедура ПриВыбореЗакладки(НомерЗакладки)
   Если НомерЗакладки = 2 тогда
      Форма.ИспользоватьСлой("ПоказанияСчетчиков, Кнопки", 2);
   Иначе
      Форма.ИспользоватьСлой("Основной, Кнопки", 2);
   КонецЕсли;
КонецПроцедуры  // ПриВыбореЗакладки
```

Наконец, процедура `ПриИзмененииВида()` активизирует закладку **Показания Водомера** и назаначается элементу интерфейса **Вода по водомеру**.

```
Процедура ПриИзмененииВида()
   Если ВидУчетаВоды = 1 тогда
      Форма.ИспользоватьЗакладки(1);
      Форма.Закладки.ДобавитьЗначение("Общий","Общие");
      Форма.Закладки.ДобавитьЗначение("ПоказанияСчетчиков", "Показа-
                                    ния счетчиков");
   Иначе
      Форма.Закладки.УдалитьЗначение(1);
      Форма.Закладки.УдалитьЗначение(1);
      Форма.ИспользоватьЗакладки(0);
```

```
    КонецЕсли;
    Форма.ИспользоватьСлой("Основной, Кнопки", 2);
КонецПроцедуры // ПриИзмененииВида
```

4. ПриЗакрытии().

Данная процедура выполняется при интерактивном закрытии формы элемента справочника. Предопределенной процедурой ПриЗаписи() отслеживается момент записи элемента справочника, но не момент окончания редактирования элемента справочника в форме. Данная процедура полезна для заполнения некоторых подчиненных справочников при вводе нового элемента.

Предопределенные процедуры формы списка справочника

Рассмотрим предопределенные процедуры форм списка справочников, которые задаются в модулях форм списка справочника соответственно:

- ПриОткрытии();
- ПриЗакрытии();
- ПриЗаписи();
- ПриВводеСтроки();
- ПриРедактированииНовойСтроки();
- ПриНачалеРедактированияСтроки();
- ПриВыбореСтроки();
- ПриПереносеЭлементаВДругуюГруппу();
- ПриВыбореРодителя();
- ПриВыбореВладельца();
- ПриСменеИерархии();
- ПриУстановкеОтбора().

1. ПриОткрытии().

Процедура ПриОткрытии() выполняется при интерактивном открытии формы списка справочника. В этой процедуре формы списка можно проанализировать настройки пользователя или атрибут формы **Параметр** для установки отбора или изменения интерфейса формы.

2. ПриЗакрытии().

Процедура ПриЗакрытии() выполняется при интерактивном закрытии формы списка справочника. Данная процедура используется для сохранения специфических настроек формы, например, показ остатков на складе:

```
Процедура ПриЗакрытии()
    СохранитьЗначение ("ФормаПодбораТоваровОтгружен-
    ных_ФлагОстаткиВКолонке", ФлагОстаткиВКолонке);
КонецПроцедуры  // ПриЗакрытии()
```

3. ПриЗаписи().

Процедура ПриЗаписи(СписокПериодическихРеквизитов) относится к записи элемента справочника в случае, когда редактирование элемента справочника производится в форме списка справочника, а не в отдельной форме.

4. ПриВводеСтроки().

Процедура ПриВводеСтроки() выполняется при интерактивном вводе новой строки списка справочника.

5. ПриРедактированииНовойСтроки().

Процедура ПриРедактированииНовойСтроки() выполняется при интерактивном редактировании новой строки списка справочника. Процедура ПриВводеСтроки() относится к вводу нового элемента справочника в случае, когда редактирование элемента справочника производится в форме списка справочника, а не в отдельной форме. Аналогична процедуре ВводНового() для формы элемента справочника.

6. ПриНачалеРедактированияСтроки().

Процедура ПриНачалеРедактированияСтроки() выполняется при начале интерактивного редактирования существующей строки списка. Процедура относится к открытию существующего элемента справочника в случае, когда редактирование элемента справочника производится в форме списка справочника, а не в отдельной форме. Аналогична процедуре ПриОткрытии() для формы элемента справочника.

7. ПриВыбореСтроки().

Процедура ПриВыбореСтроки() выполняется при выборе строки списка (двойной щелчок мыши или нажатие клавиши <Enter>).

Рассмотрим созданный ранее документ "ПоказанияВодомеров". Выбор квартиры в табличной части документа производится из справочника "Квартиры". Создадим новую форму списка для выбора квартиры и создадим в модуле этой формы списка предопределенную процедуру ПриВыбореСтроки() с использованием функции СтатусВозврата():

```
Процедура      ПриВыбореСтроки()

Если Вопрос("Вы уверены, что выбрали правильно квартиру " + Наименование,1,60)=2 тогда
                    СтатусВозврата(0);
          Возврат
          КонецЕсли;
КонецПроцедуры  // ПриВыбореСтроки()
```

После выбора пользователем квартиры из списка на экране появится запрос на подтверждение выбора (см. рис. 2.20).

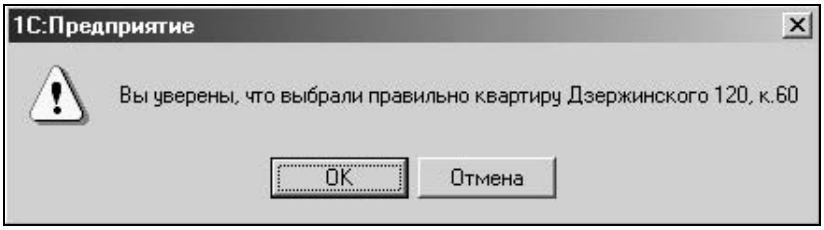


Рис. 2.20. Диалоговое окно вопроса
на подтверждение выбора строки справочника "Квартиры"

Если пользователь подтверждает свой выбор, в документ возвращается выбранная строка справочника. Если пользователь нажмет кнопку **Отмена**, процедура `ПриВыбореСтроки()` вернет его в справочник для выбора другой строки.

8. `ПриПереносеЭлементаВДругуюГруппу()`.

Процедура `ПриПереносеЭлементаВДругуюГруппу(Элемент,Группа)` выполняется при интерактивном переносе элемента справочника в другую группу.

Параметры:

- **Элемент** — элемент справочника;
- **Группа** — группа, в которую переносится элемент справочника.

Зададим в справочнике "Квартиры" 3 уровня иерархии. На первом уровне введем группы улиц, на втором — дома, на третьем — квартиры (см. рис. 2.21).

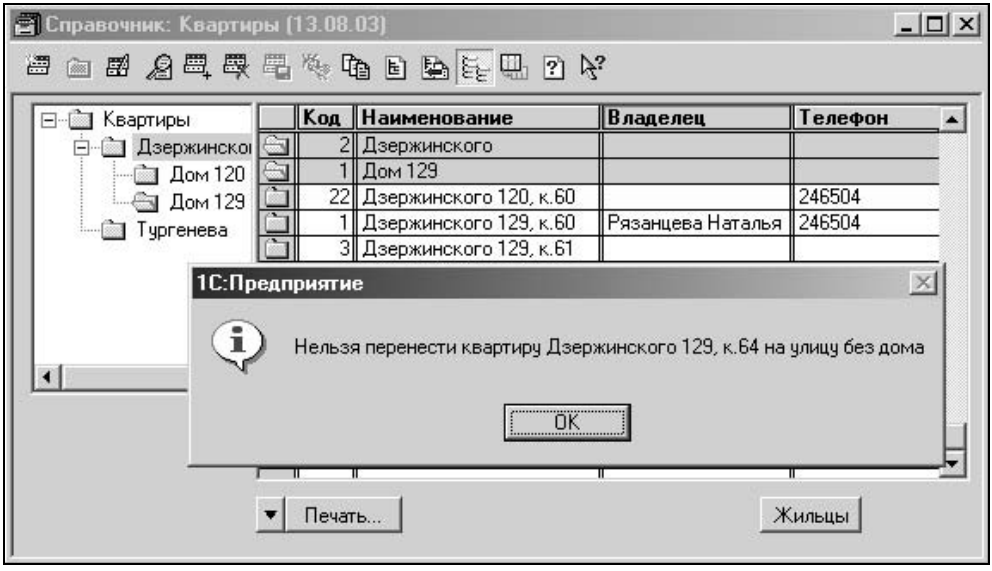


Рис. 2.21. Диалоговое окно справочника "Квартиры"

Создадим предопределенную процедуру ПриПереносеЭлементаВДругуюГруппу():

```
Процедура ПриПереносеЭлементаВДругуюГруппу(Элемент,Группа)
Если ПустаяСтрока(Группа.Родитель)=1 тогда
      Предупреждение("Нельзя перенести квартиру " + Элемент +
                     " на улицу без дома",20);
      СтатусВозврата(0);
      Возврат
  КонецЕсли;
КонецПроцедуры // ПриПереносеЭлементаВДругуюГруппу
```

При ошибочном выборе, вместо дома на втором уровне сразу улицы на первом уровне, пользователю процедурой будет выдано предупреждение и действие в соответствии со статусом возврата будет отменено.

9. ПриВыбореРодителя().

Процедура ПриВыбореРодителя(Элемент) выполняется при интерактивной смене родительской группы справочника (выбор следующего или предыдущего уровня).

Параметр — Элемент, определяющий значение элемента справочника, который интерактивно устанавливается в качестве родителя.

10. ПриВыбореВладельца().

Процедура ПриВыбореВладельца(Элемент) выполняется при интерактивном выборе владельца подчиненного справочника (при интерактивной смене владельца, т.е. смене позиции в справочнике-владельце, которая приводит к смене отображаемых в подчиненном справочнике элементов).

Параметр — Элемент, определяющий значение элемента справочника, который интерактивно устанавливается в качестве владельца подчиненного справочника.

11. ПриСменеИерархии().

Процедура ПриСменеИерархии(Способ) выполняется при интерактивной смене режима отображения иерархии справочника (пункт меню **Иерархический список**).

Параметр — Способ, определяющий значение устанавливаемого (тот, который пользователь хочет установить) способа просмотра справочника:

- 1 - иерархический список;
- 0 - все элементы сразу.

12. ПриУстановкеОтбора().

Процедура ПриУстановкеОтбора(ТипОтбора,ЗначениеОтбора) выполняется при интерактивной установке отбора любым способом (отбор, быстрый отбор, отбор по значению, история отбора) и при отключении отбора.

Параметры:

- `ТипОтбора` строковое значение, определяющее тип устанавливаемого отбора (имя реквизита справочника, по которому устанавливается отбор).
- `ЗначениеОтбора` устанавливаемое значение отбора.

Предопределенные процедуры документов

С точки зрения информации, содержимое документа можно разделить на две части:

1. Информация первичного документа, т.е. совокупность реквизитов, которые задает пользователь.
2. Бухгалтерская операция, движения регистров, расчеты в журнале расчетов, которые создаются на основании введенных реквизитов в результате проведения документа.

Предопределенные процедуры, которые выполняются при действиях пользователя с первой частью информации документа, задаются в модуле формы документа. Предопределенные процедуры, которые выполняются при проведении документа, задаются в модуле документа.

Предопределенные процедуры формы документов

Для обработки интерактивных действий пользователя с документами существуют следующие предопределенные процедуры:

- `ВводНового();`
- `ПриОткрытии();`
- `ПриЗакрытии();`
- `ПриЗаписи();`
- `ПриВводеСтроки();`
- `ВводНаОсновании();`
- `ПриРедактированииНовойСтроки();`
- `ПриНачалеРедактированияСтроки();`
- `ПриОкончанииРедактированияСтроки();`
- `ПриИзмененииПорядкаСтрок();`
- `ПриПереносеЭлементаВДругуюГруппу();`
- `ПриУдалении Строки().`

Как видно из приведенного перечня предопределенных процедур, пять первых процедур аналогичны соответствующим предопределенным процедурам справочников.

Рассмотрим специфические предопределенные процедуры, характерные только для документов.

1. ВводНаОсновании().

Процедура ВводНаОсновании(ДокументОснование) выполняется при интерактивном вводе нового документа на основании другого документа.

Параметр — ДокументОснование, определяющий значение типа Документ, на основании которого вводится новый документ.

Для иллюстрации создадим необходимые для работы бухгалтерии документы:

- "ПоступлениеОС";
- "УслугиСторонниxОрганизаций";
- справочники:
  - "Контрагенты";
  - "Договоры";
  - "ОсновныеСредства";
  - "ОбъектыСтроительства";
  - "Валюты";
  - "ВариантыРасчетаНалогов";
  - "СтавкиНДС";
  - "Банки";
  - "РасчетныеСчета";
  - "Подразделения";
  - "Сотрудники";
  - "КлассификаторЕН";
  - "ПланСчетов";
- перечисления:
  - "ПричиныВыбытияОС";
  - "СостоянияОС";
  - "СпособыНачисленияАмортизацииОС".

Оформим поступление основного средства "Комплект офисной мебели" с дополнительной услугой сторонней организации "Доставка и сборка мебели", которую введем на основании документа поступления. В документе "Услуги СторонниxОорганизаций" (см. рис. 2.22) в диалоговом окне **Вводить на основании** установим признак документа — "Поступление ОС".

Процедурой ВводНаОсновании() заполняются поля нового документа, необходимые для формирования проводок

```
Процедура ВводНаОсновании(ДокОсн)
  ДокументПоступления = ДокОсн;
```

```
    ЗачитыватьАванс = ДокОсн.ЗачитыватьАванс;
    ВариантРасчетаНалогов = ДокОсн.ВариантРасчетаНалогов;
    ТипИсполнителя = 1;
    НДСвключатьВСтоимость = ДокОсн.НДСвключатьВСтоимость;
    Комментарий = "Введен на основании:" +ГлПредставлениеДокумента(ДокОсн);
КонецПроцедуры  // ВводНаОсновании()
```

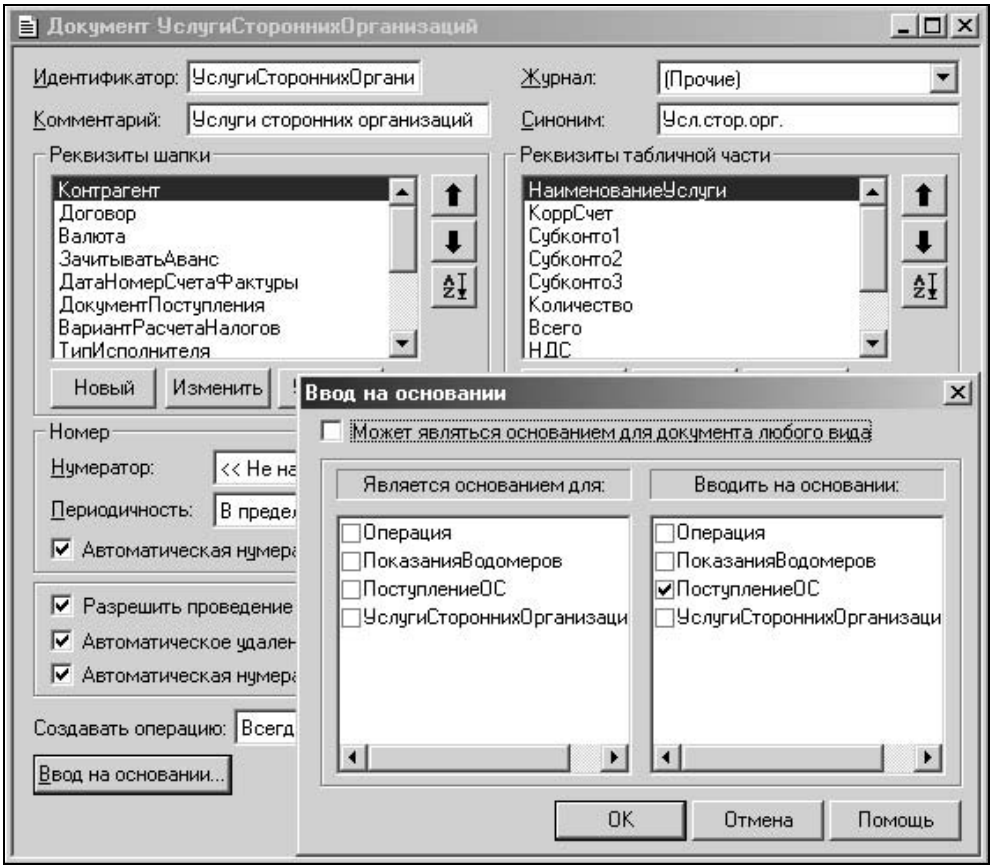


Рис. 2.22. Диалоговое окно документа "УслугиСторонниховОрганизаций"

В данном случае в модуле проведения документа реквизиты **ДокументПоступления** и **ТипИсполнителя,** заполненные процедурой `ВводНаОсновании()`, содержат информацию для формирования проводок по счету 08.4 "Приобретение отдельных объектов ОС" и счету 60.1 "Расчеты с поставщиками в рублях".

2. ПриВводеСтроки().

Процедура ПриВводеСтроки() выполняется при интерактивном вводе строки табличной части документа. Процедура может быть полезной, если по каким-либо признакам необходимо заблокировать ввод новой строки.

3. ПриНачалеРедактированияСтроки().

Процедура ПриНачалеРедактированияСтроки() выполняется при интерактивном редактировании строки табличной части документа. Процедуру удобно использовать при однотипных операциях в разных реквизитах строки.

4. ПриОкончанииРедактированияСтроки().

Процедура ПриОкончанииРедактированияСтроки(НовСтр) выполняется при окончании редактирования строки табличной части документа.

Параметры:

- НовСтр флаг новой строки:
  - 1 — если произошло окончание ввода новой строки,
  - 0 — если произошло окончание редактирования существующей строки.

5. ПриИзмененииПорядкаСтрок().

Процедура ПриИзмененииПорядкаСтрок(Действие) выполняется при изменении порядка строк табличной части документа.

Параметры — Действие:

- 1 — перемещение строки вверх;
- 1 — перемещение строки вниз;
- 0 — перенумерация строк.

6. ПриУдаленииСтроки().

Процедура ПриУдаленииСтроки() выполняется при интерактивном удалении строки табличной части документа.

Интерактивные действия пользователя, которые необходимо ограничить по каким-либо признакам, удобно обрабатывать в вышеприведенных процедурах. Рассмотрим документ "ПоказанияВодомеров". Документы начисления по квартирам вводятся ежемесячно, отчетный период — месяц. После закрытия периода документ редактировать нельзя. Обработать данный запрет можно предопределенной процедурой:

```
Процедура ПриУдаленииСтроки()
  Если ПроведенВПрошлом=1 Тогда
    Предупреждение("Документ проведен в прошлом периоде. Его нельзя
                   редактировать.",10);
    СтатусВозврата(0);
  КонецЕсли;
КонецПроцедуры  // ПриУдаленииСтроки
```

Предопределенные процедуры модуля документа

Для обработки проведения документа существуют следующие предопределенные процедуры:

- `ОбработкаПроведения()`;
- `ОбработкаУдаленияПроведения()`;
- `АрхивироватьДокумент()`.

1. `ОбработкаПроведения()`.

`ОбработкаПроведения(Знач)` — процедура обработки проведения документа в любом режиме.

Параметр — `Знач`. Это идентификатор переменной, которая получает значение из процедуры `Провести()`.

Создадим процедуру `ОбработкаПроведения()` для нашего документа "ПоказанияВодомеров". Все расчеты начислений оплаты за коммунальные услуги по квартирам будем вести в журнале расчетов "Начисления" (см. рис. 2.23).

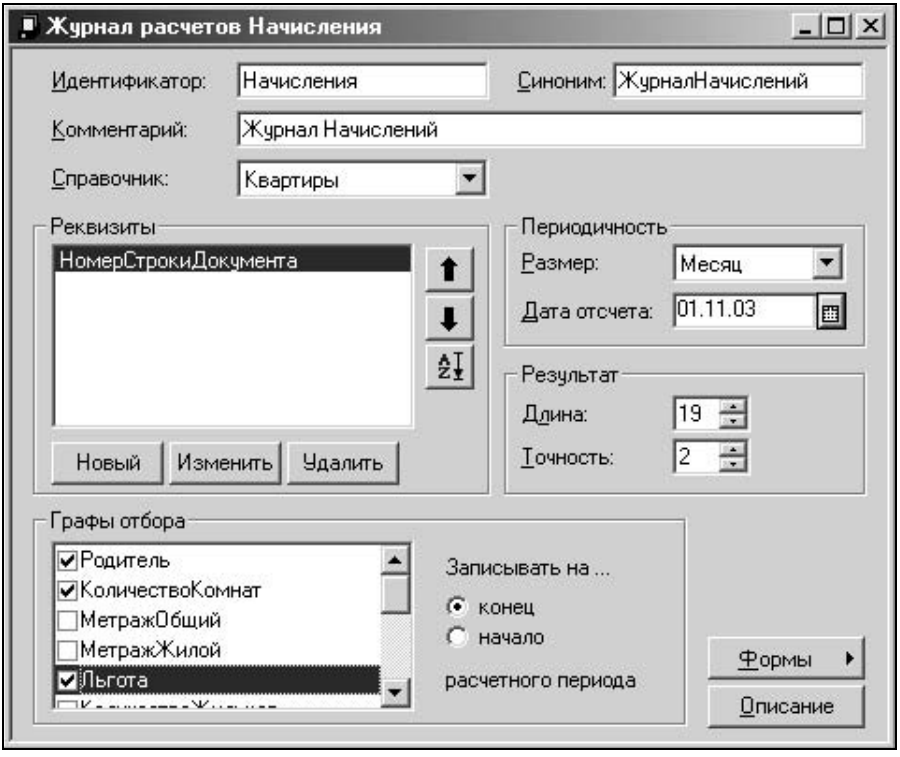


Рис. 2.23. Диалоговое окно **Журнал расчетов Начисления**

Определим вид расчета оплаты коммунальной услуги "Холодная вода по водомеру" (см. рис. 2.24).

Предоставим пользователю возможность задавать шаблоны проводок по заданным видам расчетов, т.е. создадим справочник "Виды расчетов" (см. рис. 2.25).

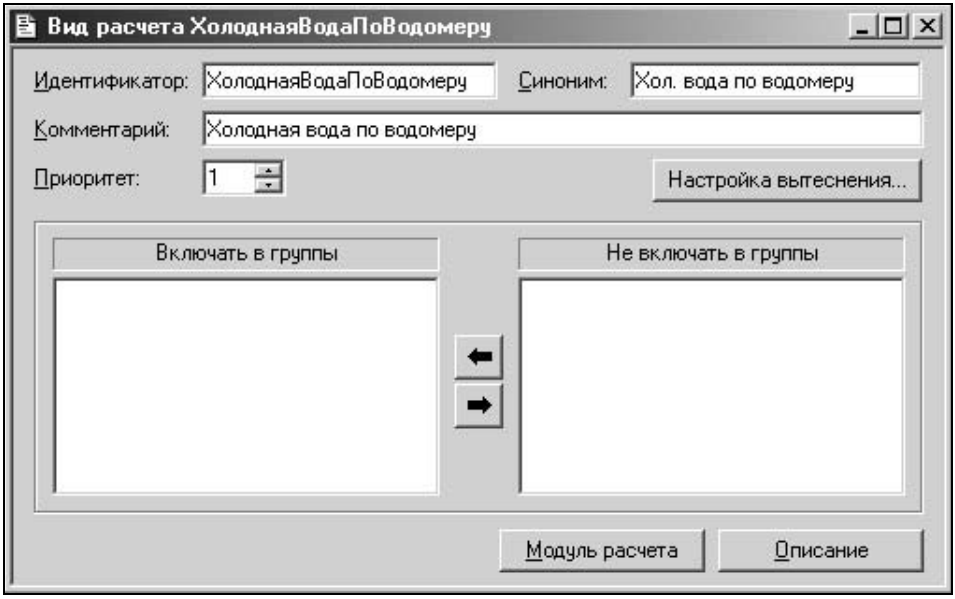


Рис. 2.24. Диалоговое окно **Вид расчета Холодная вода по водомеру**

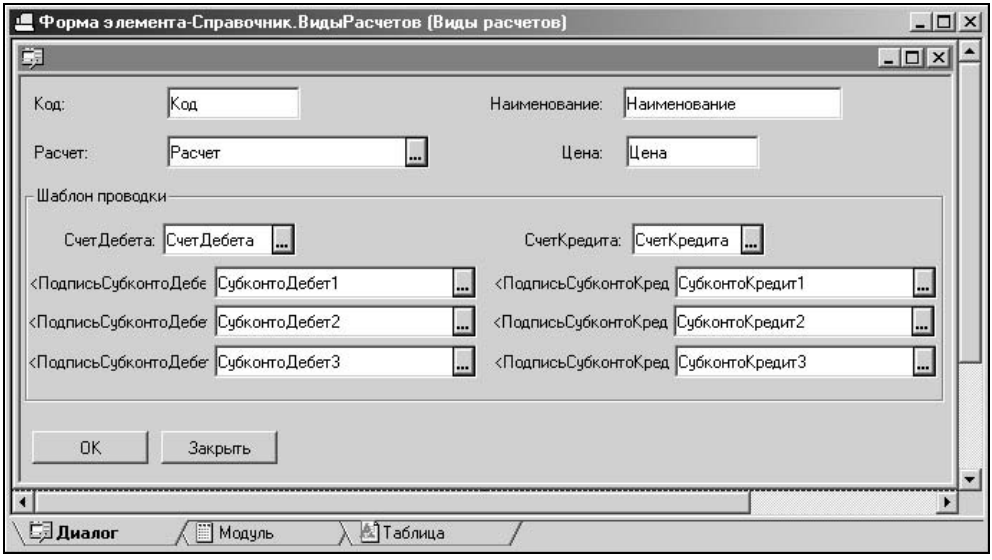


Рис. 2.25. Диалоговое окно **форма элемента-Справочник.ВидыРасчетов**

Предоставим пользователю возможность также задавать произвольные виды расчетов, т.е. создадим обработку "Ввод расчета" в виде помощника ввода нового вида расчета и введем для ее запуска кнопку **Помощник ввода расчета** (см. рис. 2.26).

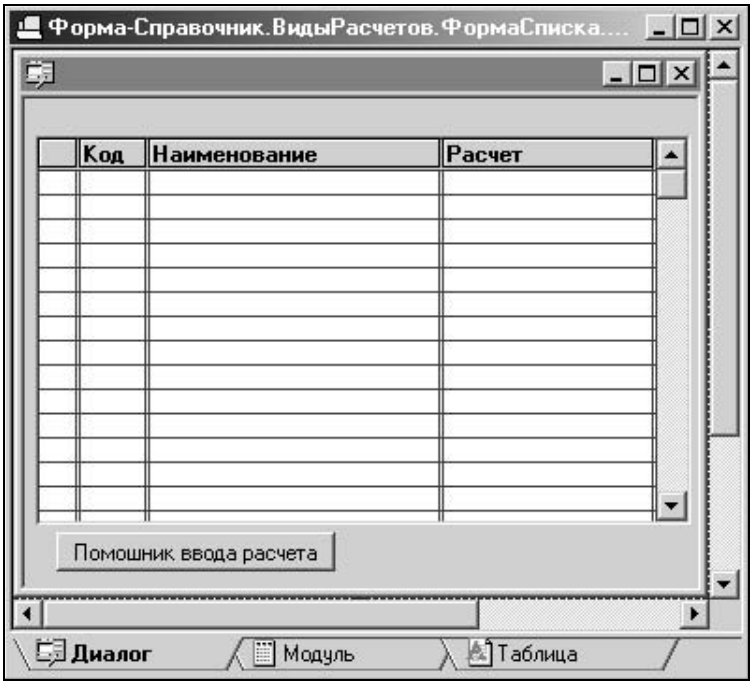


Рис. 2.26. Диалоговое окно **форма-Справочника.ВидыРасчетов**

С учетом всех необходимых расчетов определим процедуру обработки проведения:

```
Процедура ОбработкаПроведения()
  ЖурналРасчетовНачислений =
  СоздатьОбъект("ЖурналРасчетов.Начисления");
  ЖурналРасчетовНач = СоздатьОбъект("ЖурналРасчетов.Начисления");
  ДатаН = НачМесяца(ДатаДок);
  ДатаК = КонМесяца(ДатаДок);
  Пока ПолучитьСтроку()=1 Цикл
//Создадим записи в журнале расчетов
    Если  ЖурналРасчетовНачислений.ВыбратьПериодПоОбъекту(Квартира,
            ЖурналРасчетовНачислений.КонецТекущегоПериода())  =  1
            Тогда
      ЖурналРасчетовНачислений.УстановитьРеквизит
```

```
                    ("НомерСтрокиДокумента",НомерСтроки);
        ЖурналРасчетовНачислений.ВвестиРасчет
                    (Квартира,Услуга.Расчет,ДатаН,ДатаК);
    Иначе
        ЖурналРасчетовНачислений.Новая();
        ЖурналРасчетовНачислений.УстановитьРеквизит
        ("Объект",Квартира);
        ЖурналРасчетовНачислений.УстановитьРеквизит
        ("ВидРасч",Услуга.Расчет);
        ЖурналРасчетовНачислений.УстановитьРеквизит
        ("Документ", ТекущийДокумент());
        ЖурналРасчетовНачислений.УстановитьРеквизит
        ("РодительскийДокумент", ТекущийДокумент());
        ЖурналРасчетовНачислений.УстановитьРеквизит
        ("ДатаНачала",ДатаН);
        ЖурналРасчетовНачислений.УстановитьРеквизит
        ("ДатаОкончания",ДатаК);
        ЖурналРасчетовНачислений.УстановитьРеквизит
        ("НомерСтрокиДокумента", НомерСтроки);
        ЖурналРасчетовНачислений.Записать();
    КонецЕсли;
КонецЦикла;
Жильцы = СоздатьОбъект("Справочник.Жильцы");
ТаблицаНачислений = СоздатьОбъект("ТаблицаЗначений");
ТаблицаНачислений.НоваяКолонка
("Контрагент","Справочник.Контрагенты");
ТаблицаНачислений.НоваяКолонка("Сумма");
ЖурналРасчетовНач.ВыбратьЗаписиПоДокументу(ТекущийДокумент());
Пока   ЖурналРасчетовНач.ПолучитьЗапись()=1 цикл
    ЖурналРасчетовНач.Рассчитать();
    Если ЖурналРасчетовНач.Результат>0 тогда
        Жильцы.ИспользоватьВладельца(ЖурналРасчетовНач.Объект);
        Жильцы.ВыбратьЭлементы();
        Пока Жильцы.ПолучитьЭлемент()=1 цикл
          Если Жильцы.СтатусПрописки=Перечисление.Статусы.Владелец
тогда
            ТаблицаНачислений.НоваяСтрока();
            ТаблицаНачислений.Контрагент = Жильцы.Контрагент;
            ТаблицаНачислений.Сумма = ЖурналРасчетовНач.Результат;
```

```
                Прервать;
            КонецЕсли;
        КонецЦикла;
    КонецЕсли;
КонецЦикла;
ТаблицаНачислений.Свернуть("Контрагент","Сумма");
ТаблицаНачислений.ВыбратьСтроки();
// Создадим проводки
Пока ТаблицаНачислений.ПолучитьСтроку()=1 Цикл
    Операция.НоваяПроводка();
    Операция.Кредит.Счет = СчетПоКоду("90.1.1");
    Операция.Кредит.ВидыНоменклатуры = Констан-
                                    та.ОсновнойВидНоменклатуры;
    Операция.Дебет.Счет = СчетПоКоду("62.1");
    Операция.Дебет.Контрагенты = ТаблицаНачислений.Контрагент;
    Операция.Сумма = ТаблицаНачислений.Сумма;
    Операция.СодержаниеПроводки = "Начисл. за ком. услуги";
КонецЦикла;
Операция.Записать();
КонецПроцедуры
```

2. `ОбработкаУдаленияПроведения()`.

Процедура `ОбработкаУдаленияПроведения()` выполняется при обработки удаления проведения документа или удаления проведенных документов в любом режиме. Если некоторые реквизиты получены расчетным путем с использованием периодических реквизитов документов или справочников, результат расчета может остаться некорректным даже после простого удаления проведения. Для выполнения поправок и существует предопределенная процедура `ОбработкаУдаленияПроведения()`. Уберем расчет суммы оплаты воды по водомеру в нашем документе "ПоказанияВодомеров":

```
Процедура ОбработкаУдаленияПроведения()
  Если ВыбратьСтроки() = 1 Тогда
     Пока ПолучитьСтроку() = 1 Цикл
        Сумма = 0;
     КонецЦикла;
  КонецЕсли;
КонецПроцедуры  // ОбработкаУдаленияПроведения()
```

3. `АрхивироватьДокумент()`.

Процедура `АрхивироватьДокумент()` выполняется для обработки архивирования документа.

Данная процедура может располагаться только в модуле документа, вызываемом в Конфигураторе — **Документ | Редактировать | Модуль документа**. Вызов процедуры происходит в процессе смены расчетного периода журнала расчетов.

Предопределенные процедуры журнала документов

Предопределенных процедуры журналов документов задаются в модуле формы журнала документов. Для обработки интерактивных действий пользователя с журналом документов существуют следующие предопределенные процедуры:

- ПриУстановкеОтбора();
- ПриВыбореСтроки();
- ПриУстановкеИнтервала().

1. ПриУстановкеОтбора().

Процедура ПриУстановкеОтбора(ИмяРеквизОтбора,Значение) выполняется при интерактивной установке отбора документов в журнале.

Параметры:

- ИмяРеквизОтбора — строка с названием общего реквизита документа, по которому производится отбор;
- Значение — значение реквизита отбора.

2. ПриВыбореСтроки().

Процедура ПриВыбореСтроки() выполняется при выборе строки списка (двойной щелчок кнопкой мыши или нажать клавишу <Enter>). Целесообразно использовать данную процедуру при дальнейшем использовании выбранного документа журнала:

```
Процедура ПриВыбореСтроки()
    Форма.Параметр = ТекущийДокумент;
КонецПроцедуры    // ПриВыбореСтроки()
```

3. ПриУстановкеИнтервала().

Процедура ПриУстановкеИнтервала(ДатаНач,ДатаКон) выполняется при установке интервала журнала.

Параметры:

- ДатаНач — дата начала интервала журнала;
- ДатаКон дата конца интервала журнала.

В нашей конфигурации в журнале **Операция** в модуле формы списка создадим процедуру:

```
Процедура ПриУстановкеИнтервала(ДатаНач, ДатаКон)
```

```
    Если (ДатаНач < ДатаИзВарианта(НачалоСтандартногоИнтервала(), 0))
        или (ДатаКон>ДатаИзВарианта(КонецСтандартногоИнтервала(), 1))
        Тогда

      Предупреждение("Указанный интервал выходит за границы,
      |установленные в параметрах системы для журналов документов,
      |операций и проводок. Он не сохранится после закрытия журнала.

      |Установка интервала для всех журналов выполняется из меню Сервис
      в |настройке параметров системы (Параметры) на закладке Журналы.");

    КонецЕсли;

КонецПроцедуры
```

В процедуре используется функция определения дат начала и конца интервала из настройки пользователя `ДатаИзВарианта()`:

```
Функция ДатаИзВарианта(Вариант, КонецИнтервала)
  ДатаГраницы = "";
  Если ТипЗначенияСтр(Вариант) = "Дата" Тогда
     ДатаГраницы = Вариант;
  Иначе
     Если Вариант = "Месяц" Тогда
        Если КонецИнтервала = 0 Тогда //начало интервала
           ДатаГраницы = НачМесяца(РабочаяДата());
        Иначе
           ДатаГраницы = КонМесяца(РабочаяДата());
        КонецЕсли;
     ИначеЕсли Вариант = "Квартал" Тогда
        Если КонецИнтервала = 0 Тогда //начало интервала
           ДатаГраницы = НачКвартала(РабочаяДата());
        Иначе
           ДатаГраницы = КонКвартала(РабочаяДата());
        КонецЕсли;
     ИначеЕсли Вариант = "Год" Тогда
        Если КонецИнтервала = 0 Тогда //начало интервала
           ДатаГраницы = НачГода(РабочаяДата());
        Иначе
           ДатаГраницы = КонГода(РабочаяДата());
        КонецЕсли;
     Иначе
        ДатаГраницы = РабочаяДата();
    КонецЕсли;
  КонецЕсли;
  Возврат ДатаГраницы;
КонецФункции  //ДатаИзВарианта
```

Внимание

Приведенная процедура `ПриУстановкеИнтервала()` и функция `ДатаИзВарианта()` разработаны в типовой конфигурации "1С:Бухгалтерский учет".

Предопределенные процедуры журнала расчетов

Предопределенные процедуры журналов расчетов задаются в модуле формы журнала расчетов. Для обработки интерактивных действий пользователя с журналом документов существуют следующие предопределенные процедуры:

- `ПриИсправленииРезультата()`;
- `ПриОтменеИсправления()`;
- `ПриРасчете()`;
- `ПриУстановкеОтбора()`;
- `ПриУстановкеГраницыПросмотра()`;
- `ПриУстановкеПредставления()`;
- `ПриВыбореВладельца()`.

1. `ПриИсправленииРезультата()`.

Процедура `ПриИсправленииРезультата(Запись)` выполняется при редактировании результата расчета записи журнала расчетов.

Параметр — `Запись` является ссылкой на запись журнала расчетов, результат расчета которой исправляется.

Если результаты расчета по какой-либо причине, например, убрать копейки, необходимо отредактировать, то по двойному щелчку кнопки мыши вызывается диалоговое окно **Исправление результата.** После ввода нужной суммы, выполняется процедура `ПриИсправленииРезультата()`.

Создадим процедуру в нашем журнале расчетов "Начисления" с запросом на подтверждение:

```
Процедура ПриИсправленииРезультата(Запись)
  Если Запись.Исправлена = 0 тогда
    Если Вопрос("Сохранить изменения результата расчета?",1,60)=2
тогда
      СтатусВозврата(0);
      Возврат;
    КонецЕсли;
  КонецЕсли;
КонецПроцедуры  // ПриИсправленииРезультата()
```

2. ПриОтменеИсправления().

Процедура ПриОтменеИсправления(Запись) выполняется при отмене исправления результата расчета в журнале расчета.

Параметр — Запись является ссылкой на запись журнала расчетов, исправление результата которой отменяется.

Если необходимо отменить ручное исправление и рассчитать запись заново, то двойным щелчком кнопки мыши вызывается диалоговое окно **Исправление результата.** После ввода в этом окне нужной суммы, выполняется процедура ПриИсправленииРезультата().

Создадим процедуру в журнале расчетов "Начисления" с запросом на подтверждение:

```
Процедура ПриОтменеИсправления(Запись)
  Если Вопрос("Исправление будет потеряно! Отменить исправление?",1,60)=2 тогда
     СтатусВозврата(0);
     Возврат;
  КонецЕсли;
КонецПроцедуры  // ПриОтменеИсправления()
```

3. ПриРасчете().

Процедура ПриРасчете(ОбъектРасчета) выполняется при расчете одной записи, записей объекта или записей документа в журнале расчетов.

Параметр — ОбъектРасчета представляется записью журнала расчетов или элемент справочника, являющийся объектом расчета, или документ, записи которого рассчитываются. Что в данный момент передается системой при вызове процедуры, зависит от выполняемого действия — расчет одной записи, расчет объекта или расчет документа.

Для иллюстрации возможностей процедуры при расчете по всему объекту, введем новый документ "Ввод начального сальдо" (см. рис. 2.27).

Введем константу ПределСуммыДолга, чтобы предоставить пользователю возможность задавать нижнюю границу долга за коммунальные услуги по квартире. Процедура ПриРасчете() при выборе пункта меню **Рассчитать объект** будет вести анализ задолженности квартиры за коммунальные услуги:

```
Процедура ПриРасчете(Объект)
 Если ТипЗначенияСтр(Объект ) = "Справочник" Тогда // расчет по объекту
  ЖурналРасчетов = СоздатьОбъект("ЖурналРасчетов.Начисления");
  ЖурналРасчетов.ВыбратьЗаписиПоОбъекту
                          (Объект,ДатаНачала,ДатаОкончания);
  СуммаНачисления=0;
   СуммаДолга = 0;
   Пока ЖурналРасчетов.ПолучитьЗапись()=1 Цикл
```

```
        Если ЖурналРасчетов.ВидРасч = ВидРасчета.НачальноеСальдо тогда
                СуммаДолга = ЖурналРасчетов.Результат;
        Иначе
        СуммаНачисления = СуммаНачисления+ЖурналРасчетов.Результат;
        КонецЕсли;
        КонецЦикла;
    Долг = -(СуммаНачисления+СуммаДолга);
        Если Долг>Константа.ПределСуммыДолга.Получить(ДатаОкончания)
        тогда
        Если Вопрос(Строка(Долг)+" превышает порог долга! Печатать
        сводную квитанцию?",1,60)=1 тогда
                ПечатьСводнойКвитанции();
        КонецЕсли;
    КонецЕсли;
КонецЕсли;
КонецПроцедуры
```

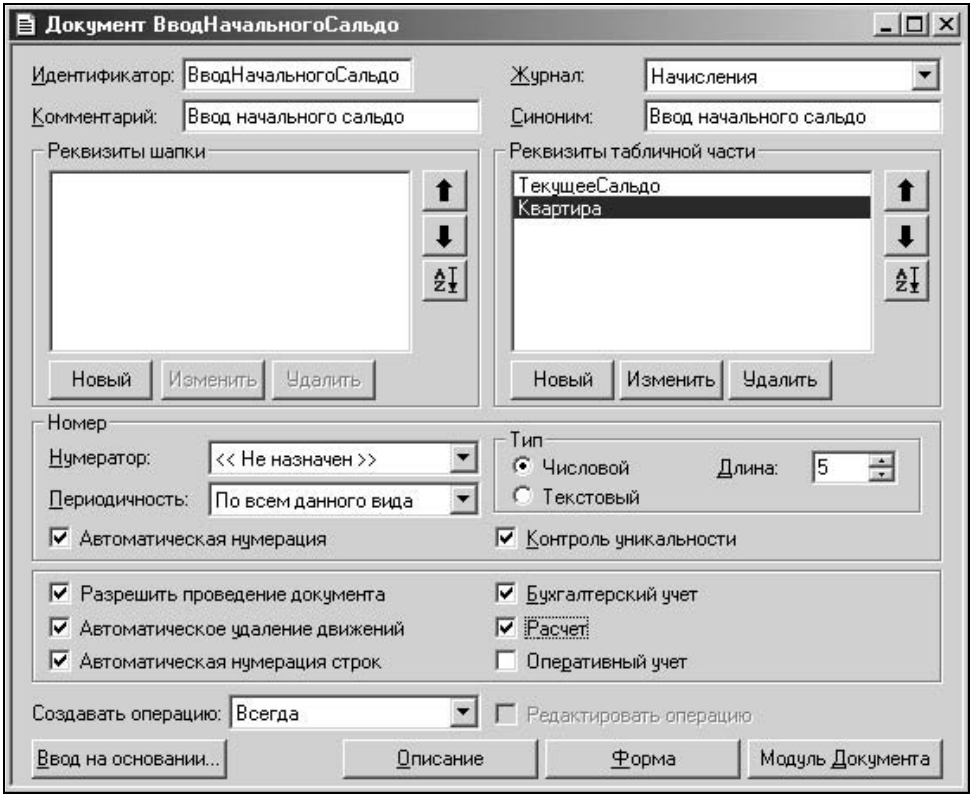


Рис. 2.27. Диалоговое окно задания документа "Ввод начального сальдо"

4. `ПриУстановкеОтбора()`.

Процедура `ПриУстановкеОтбора(ИмяОтбора,ЗначениеОтбора)` выполняется при установке отбора в форме журнала расчетов.

Параметры:

- `ИмяОтбора` — строковое значение, представляющее наименование графы отбора, которая выбрана пользователем при интерактивной установке отбора;
- `ЗначениеОтбора` — значение отбора, которое выбрано пользователем при интерактивной установке отбора.

Создадим процедуру `ПриУстановкеОтбора()` для организации отображения полной строки сложного отбора с вложенным условием, например, все записи должников по трехкомнатным квартирам заданного дома. Стандартная реализация этого такого отбора отображает в заголовке формы только последнее условие отбора, что не совсем удобно для пользователя.

Введем в начале модуле переменную `Отбор`:

```
ПеремОтбор;
```

Инициализируем значение данной переменной в конце модуля формы списка:

```
Отбор ="";
```

Наша процедура выполняется до того, как меняется заголовок формы, поэтому в форме списка зададим элемент диалога типа **Текст** с идентификатором **ТекстОтбора** (см. рис. 2.28).

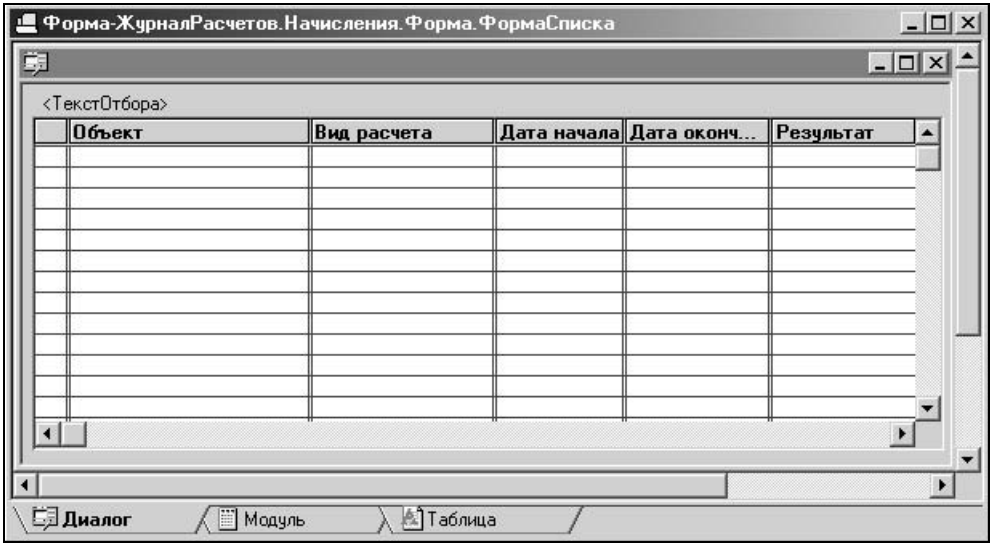


Рис. 2.28. Диалоговое окно формы журнала "Начисления"

Наша процедура формирует полную строку установленного отбора:

```
Процедура ПриУстановкеОтбора(ИмяОтбора,ЗначениеОтбора)
  Если ПустаяСтрока(ИмяОтбора)=1 тогда
     Отбор="";
  ИначеЕсли ПустаяСтрока(Отбор)=1 тогда
     Отбор = "Установлен отбор "+ИмяОтбора+" = "+ЗначениеОтбора;
  Иначе
     Отбор=Отбор+" и "+ИмяОтбора+" = "+ЗначениеОтбора;
  КонецЕсли;
  Форма.ТекстОтбора.Заголовок(Отбор);
КонецПроцедуры
```

5. `ПриУстановкеГраницыПросмотра()`.

Процедура `ПриУстановкеГраницыПросмотра(Период)` выполняется при установке границы просмотра журнала расчетов.

Параметр — `Период` определяет значение типа "Период журнала расчетов". `Период`, устанавливается пользователем как граница просмотра записей.

Аналогично процедуре `ПриУстановкеОтбора()` можно отслеживать отображение границы просмотра записей процедурой `ПриУстановкеГраницыПросмотра()`, хотя это не так актуально.

6. `ПриУстановкеПредставления()`.

Процедура `ПриУстановкеПредставления(Режим)` выполняется при установке режима представления записей журнала расчетов.

Параметр — `Режим` представляет числовое значение или режим представления записей журнала расчетов, устанавливаемый пользователем. Параметр может принимать следующие значения:

- 1 — выбран режим представления всех записей журнала расчетов за текущий период;
- 2 — выбран режим представления записей текущего объекта за текущий период;
- 3 — выбран режим представления записей текущего документа за текущий период.

Аналогично процедуре `ПриУстановкеОтбора()` можно отслеживать отображение текущей суммы долга при выполнении процедуры `ПриУстановкеПредставления()`, с параметром 2. Например:

```
Процедура ПриУстановкеПредставления(Режим)
  Если Режим = 2 тогда
    ЖурналРасчетов = СоздатьОбъект("ЖурналРасчетов.Начисления");
    ЖурналРасчетов.ВыбратьЗаписиПоОбъекту
```

```
(Объект,ДатаНачала,ДатаОкончания);
 СуммаНачисления=0;
СуммаДолга = 0;
Пока ЖурналРасчетов.ПолучитьЗапись()=1 Цикл
   Если ЖурналРасчетов.ВидРасч = ВидРасчета.НачальноеСальдо тогда
         СуммаДолга = ЖурналРасчетов.Результат;
   Иначе
   СуммаНачисления = СуммаНачисления+ЖурналРасчетов.Результат;
   КонецЕсли;
КонецЦикла;
Долг = -(СуммаНачисления+СуммаДолга);
Если Долг>Константа.ПределСуммыДолга тогда
   Форма.ТекстОтбора.Заголовок
                  ("Квартира - должник! Сумма долга - "+Долг);
Иначе
   Форма.ТекстОтбора.Заголовок("");
КонецЕсли;
Иначе
  Форма.ТекстОтбора.Заголовок("");
КонецЕсли;
КонецПроцедуры
```

7. `ПриВыбореВладельца()`

Процедура `ПриВыбореВладельца(Элемент)` выбора элемента справочника, по которому будут выведены расчеты, выполняется при интерактивной смене владельца, т. е. смене позиции в "главном" справочнике, для которого создан журнал расчетов.

Параметр — `Элемент` определяет значение устанавливаемого владельца (т. е. элемент справочника, по которому будут выведены расчеты).

Предопределенные процедуры контекста формы

Формы, которые открываются не интерактивно пользователем, а вызовом из программных модулей, также имеют ряд предопределенных процедур, которые описываются в модуле формы и выполняются независимо от принадлежности к объекту метаданных:

- `ПриОткрытии()`;
- `ПриПовторномОткрытии()`;
- `ПриЗакрытии()`;
- `ПриВыбореЗакладки()`;

- ОбработкаПодбора();
- ПриНачалеВыбораЗначения();
- ОбработкаВыбораЗначения();
- ПриВыбореСтроки().

1. ПриПовторномОткрытии().

Синтаксис — ПриПовторномОткрытии().

Данная процедура выполняется при повторном открытии формы в случае, если открывают уже открытую форму (т. о. форма просто активизируется).

2. ПриВыбореЗакладки().

Процедура ПриВыбореЗакладки(НомерЗакладки,ЗначениеЗакладки) обработки подбора значения выполняется в момент интерактивного выбора пользователем закладки в форме.

Параметры:

- НомерЗакладки — числовое значение — номер выбранной закладки формы;
- ЗначениеЗакладки — значение выбранной закладки формы.

Процедуру ПриВыбореЗакладки() мы уже создали для формы элемента справочника "Квартиры" *(см. разд. "Предопределенные процедуры формы элемента справочника")*. Процедура может иметь следующий вид:

```
Процедура ПриВыбореЗакладки(НомерЗакладки)
  Если НомерЗакладки = 2 тогда
     Форма.ИспользоватьСлой("ПоказанияСчетчиков, Кнопки", 2);
  Иначе
     Форма.ИспользоватьСлой("Основной, Кнопки", 2);
  КонецЕсли;
КонецПроцедуры  // ПриВыбореЗакладки
```

3. ОбработкаПодбора().

Процедура ОбработкаПодбора(Элемент,КонтФормы) обработки подбора значения выполняется нажатием кнопки **Выбрать** в **Форме Подбора** значения.

Параметры:

- Элемент — элемент справочника подбора или документ, передаваемый для обработки;
- КонтФормы — контекст той формы, из которой шел подбор.

В процедуре ОбработкаПодбора() программист может обработать список выбранных элементов на предмет повтора или задания дополнительного критерия выборки из списка.

4. `ПриНачалеВыбораЗначения()`.

Процедура `ПриНачалеВыбораЗначения(ИдентЭлемДиалога,ФлагСтандОбр)` выполняется после выбора значения в форме выбора (выбор может быть инициирован в немодальном режиме интерактивно, при помощи элемента диалога с "педалькой".

Параметры:

- `ИдентЭлемДиалога` — идентификатор элемента диалога, которым инициализирован выбор значения;
- `ФлагСтандОбр` — флаг, который изначально, при вызове процедуры, равен 1 (если в теле процедуры значение этого параметра поменять на 0, то стандартный процесс выбора значения не будет происходить).

5. `ОбработкаВыбораЗначения()`.

Синтаксис:

`ОбработкаВыбораЗначения(ВыбЗнач,ИдентЭлемДиалога,ФлагСтандОбр)`. Процедура выполняется после выбора значения в форме выбора (выбор может быть инициирован в немодальном режиме интерактивно, при помощи элемента диалога с "педалькой").

Параметры:

- `ВыбЗнач` — выбранный элемент справочника, документ или иной объект, передаваемый для обработки;
- `ИдентЭлемДиалога` — идентификатор элемента диалога, которым инициализирован выбор значения;
- `ФлагСтандОбр` — флаг, установка которого в теле процедуры в 0 (ноль) приведет к отмене присвоения стандартного значения.

Предопределенные процедуры форм отчетов и обработок имеют свою специфику, хоть и имеют одинаковые названия с аналогичными процедурами других объектов метаданных:

- `ВводНового()`;
- `ПриОткрытии()`;
- `ПриВыбореЯчейкиТаблицы()`.

1. `ВводНового()`.

Процедура `ВводНового()` выполняется при открытии формы отчета (обработки) и при восстановлении сохраненной настройки отчета (обработки).

2. `ПриВыбореЯчейкиТаблицы()`.

Процедура обработки ячейки таблицы в режиме ввода данных `ПриВыбореЯчейкиТаблицы(Адрес,Значение)` выполняется по двойному щелчку клавиши мыши или по нажатию клавиши <Enter> в табличном документе на выбранной ячейке.

Параметры:

- Адрес — здесь в процедуру системой передается строковое значение имени области таблицы, если выбранная ячейка помечена в таблице как отдельная область, или адрес ячейки в формате "R1C1:R2C2";
- Значение — здесь в процедуру системой передается значение данной ячейки, а если диаграмма, то передается значение выбранного элемента.

3. ПриОткрытии().

Процедура ПриОткрытии(ФлагЧтенияНастройки) выполняется при открытии формы отчета (обработки).

Параметр — ФлагЧтенияНастройки является числовым значением и определяет признак считывания сохраненной настройки отчета (обработки). Он может принимать значения:

- 1 — при открытии формы была восстановлена последняя сохраненная настройка отчета (обработки);
- 0 — при открытии формы настройка не восстановлена.

Создадим механизм настройки колонок для нашего справочника "Квартиры". При реализации его необходим не интерактивный вызов формы НастройкаКолонок.

Для этого добавим кнопку **Колонки** в форме списка справочника "Квартиры" и зададим процедуру — УстановитьВидимостьКолонок(), которая будет вызываться нажатием кнопки.

```
Процедура УстановитьВидимостьКолонок(ПриОткрытии=0)
    Перем СписокКолонок, ТекущаяКолонка, ИдКолонки, Колонка;
    ПолныйСписокКолонок = СоздатьОбъект("СписокЗначений");
    ПолныйСписокКолонок.ДобавитьЗначение("Код", "Код");
    ПолныйСписокКолонок.ДобавитьЗначение("Наименование", "Наименование");
    ПолныйСписокКолонок.ДобавитьЗначение("КоличествоКомнат", "Количе-
    ствоКомнат");
    ПолныйСписокКолонок.ДобавитьЗначение("МетражОбщий", "МетражОбщий");
    ПолныйСписокКолонок.ДобавитьЗначение("МетражЖилой", "МетражЖилой");
    ПолныйСписокКолонок.ДобавитьЗначение("Льгота", "Льгота");
    ПолныйСписокКолонок.ДобавитьЗначение("КоличествоЖильцов",
    "КоличествоЖильцов");
    ПолныйСписокКолонок.ДобавитьЗначение("Дом", "Дом");
    ПолныйСписокКолонок.ДобавитьЗначение("ДоговорПриобретения",
    "ДоговорПриобретения");
    ПолныйСписокКолонок.ДобавитьЗначение("Телефон", "Телефон");
// При первом запуске, пока пользователь не настроил видимость колонок,
// будут видимы только колонки "Код","Наименование"
```

```
ПолныйСписокКолонок.Пометка(1, 1);
ПолныйСписокКолонок.Пометка(2,        1);
Если ПриОткрытии=0 Тогда
   СписокПараметров= СоздатьОбъект("СписокЗначений");
   СписокПараметров.ДобавитьЗначение(Контекст);
   СписокПараметров.ДобавитьЗначение
   ("Справочник_Квартиры_ФормаСписка_ДляВыбора");
   СписокПараметров.ДобавитьЗначение(ПолныйСписокКолонок);
  ОткрытьФормуМодально("Обработка.НастройкаКолонок",
  СписокПараметров);
Иначе
   СписокКолонок   =      ВосстановитьЗначение ("СписокКолонок_
   Справочник_Квартиры_ФормаСписка_ДляВыбора");
   ТекущаяКолонка =      ВосстановитьЗначение ("ТекущаяКолонка_
   Справочник_Квартиры_ФормаСписка_ДляВыбора");
   Если ПустоеЗначение(СписокКолонок) = 1 Тогда
      ПолныйСписокКолонок.Выгрузить(СписокКолонок);
      СохранитьЗначение        ("СписокКолонок_Справочник_Квартиры_
      ФормаСпискаДляВыбора", СписокКолонок);
   ИначеЕсли ТипЗначенияСтр(СписокКолонок) = "СписокЗначений" Тогда
      Если СписокКолонок.РазмерСписка() <> ПолныйСписокКолонок.
          РазмерСписка() Тогда
          ПолныйСписокКолонок.Выгрузить(СписокКолонок);
          СохранитьЗначение    ("СписокКолонок_Справочник_Квартиры_
          ФормаСписка_ДляВыбора",СписокКолонок);
      КонецЕсли;
   КонецЕсли;
   Для Сч = 1 По СписокКолонок.РазмерСписка() Цикл
      ИдКолонки   =      СписокКолонок.ПолучитьЗначение(Сч);
      Попытка
         Колонка  = Форма.ПолучитьАтрибут(ИдКолонки);
      Исключение
         Сообщить(ИдКолонки);
      КонецПопытки;
      Если СписокКолонок.Пометка(Сч) = 1 Тогда
         Колонка.Видимость(1);
         Если ИдКолонки = ТекущаяКолонка Тогда
            Форма.Активизировать(ИдКолонки,0);
         КонецЕсли;
      Иначе
```

```
            Колонка.Видимость(0);
        КонецЕсли;
    КонецЦикла;
КонецЕсли;
КонецПроцедуры // УстановитьВидимостьКолонок
```

Процедура вызывается нажатием кнопки без параметров, поэтому выполняется сегмент с параметром `ПриОткрытии = 0` (см. рис. 2.29).

Создадим обработку "НастройкаКолонок" с тремя слоями (см. рис. 2.30).

Слой **Основной** содержит только кнопки (см. рис. 2.31).

Назначим процедуру `Принять(Список)` кнопке **OK** (см. рис. 2.32).

Слой **Активизировать** содержит список всех колонок (см. рис. 2.33), который передается в форму из процедуры `УстановитьВидимостьКолонок()`.

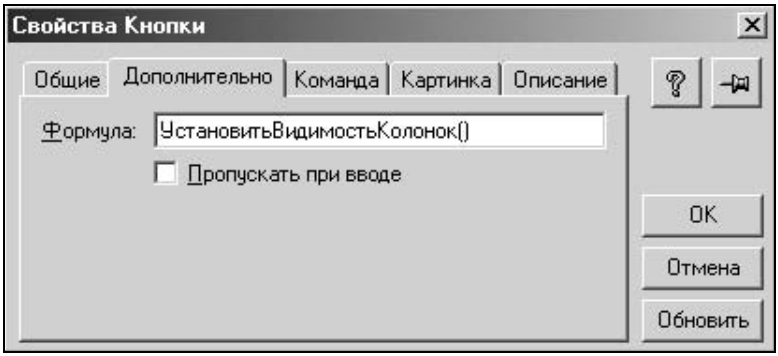


Рис. 2.29. Диалоговое окно свойства кнопки **Колонки** на вкладке **Дополнительно**

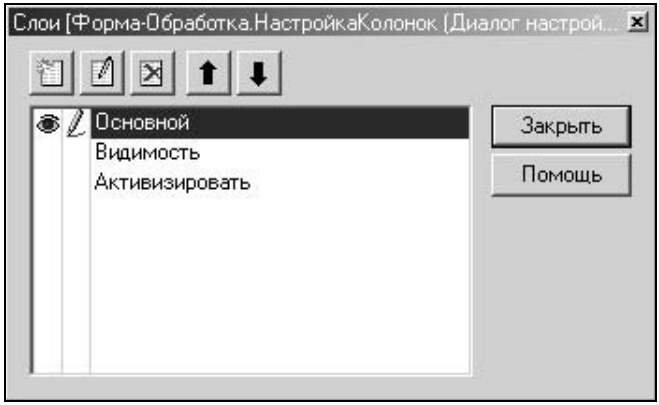


Рис. 2.30. Диалоговое окно слоев обработки "НастройкаКолонок"

Рис. 2.31. Диалоговое окно обработки "НастройкаКолонок" (слой **Основной**)

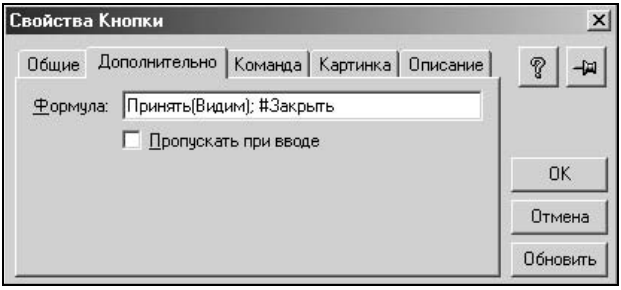


Рис. 2.32. Диалоговое окно свойств кнопки **OK** обработки "НастройкаКолонок" (слой **Основной**)

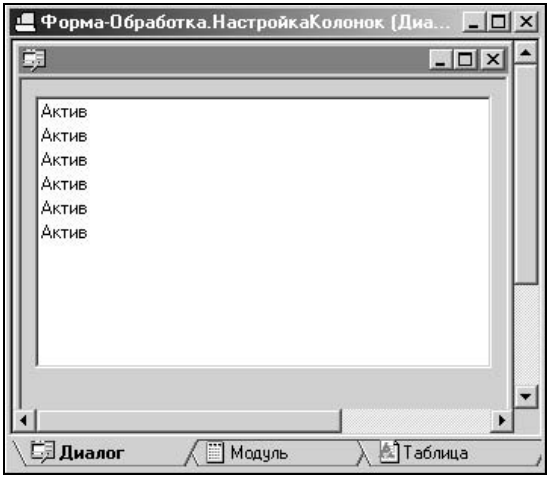


Рис. 2.33. Диалоговое окно обработки "НастройкаКолонок" (слой **Активизировать**)

Слой **Видимость** содержит список всех отмеченных колонок (см. рис. 2.34), который задается пользователем. Колонки этого списка будут видимыми в форме после нажатия кнопки **ОК**.

Рис. 2.34. Диалоговое окно обработки "НастройкаКолонок" (слой **Видимость**)

В модуле формы — НастройкаКолонок определим процедуру ПриОткрытии(), которая будет выполняться после вызова метода — ОткрытьФормуМодально(Обработка.НастройкаКолонок,СписокПараметров) модуля справочника "Квартиры":

```
Перем СписокКолонок, ТекущаяКолонка, ИдКолонки, Колонка;
Перем Конт, ИдентификаторФормы, НачальныйСписок, ПриОткрытии;
Перем ВремТекКолонка;
Процедура ПриОткрытии()
    Если ПустоеЗначение(Форма.Параметр)=1 Тогда
      Возврат;
    КонецЕсли;
    Конт = Форма.Параметр.ПолучитьЗначение(1);
    ИдентификаторФормы = Форма.Параметр.ПолучитьЗначение(2);
    НачальныйСписок = Форма.Параметр.ПолучитьЗначение(3);
    СписокКолонок=ВосстановитьЗначение("СписокКолонок_" +
                  ИдентификаторФормы);
    ТекущаяКолонка=ВосстановитьЗначение("ТекущаяКолонока_" +
                  ИдентификаторФормы);
```

```
    Если ПустоеЗначение(СписокКолонок) = 1 Тогда
      НачальныйСписок.Выгрузить(СписокКолонок);
      СохранитьЗначение ("СписокКолонок_" + ИдентификаторФормы,
                         СписокКолонок);
    ИначеЕсли ТипЗначенияСтр(СписокКолонок) = "СписокЗначений" Тогда
        Если СписокКолонок.РазмерСписка() <>
        НачальныйСписок.РазмерСписка()
      Тогда
        НачальныйСписок.Выгрузить(СписокКолонок);
        СохранитьЗначение("СписокКолонок_" +
                          ИдентификаторФормы,СписокКолонок);
      КонецЕсли;
    КонецЕсли;
    Форма.Заголовок("Настройка колонок",0);
    Форма.ИспользоватьЗакладки(1);
    Форма.Закладки.ДобавитьЗначение("Видимость","Видимость");
    Форма.Закладки.ДобавитьЗначение("Активизировать","Активизировать
                                    при открытии");
    Форма.ИспользоватьСлой("Основной,Видимость",2);
    СписокКолонок.Выгрузить(Видим);
    ВремТекКолонка = Видим.ПолучитьЗначение (макс
                     (Видим.НайтиЗначение (ТекущаяКолонка),1));
КонецПроцедуры // ПриОткрытии()
```

Внимание

Следует помнить, что переменные, которые используется во всем теле модуля в разных процедурах, определяются в начале модуля до определения первой процедуры.

Определим процедуру сохранения настройки `Принять()`:

```
Процедура Принять(Список);
    Если Актив.ТекущаяСтрока()>0 Тогда
                  ВремТекКолонка=Актив.ПолучитьЗначение
                          (Макс (Актив.ТекущаяСтрока(),1));
    КонецЕсли;
    Если ПустоеЗначение(ВремТекКолонка)=0 Тогда
      ТекущаяКолонка = ВремТекКолонка;
    КонецЕсли;
    Для Сч = 1 По Список.РазмерСписка() Цикл
      ИдКолонки = Список.ПолучитьЗначение(Сч);
```

```
        Попытка
            Колонка = Конт.Форма.ПолучитьАтрибут(ИдКолонки);
        Исключение
            Сообщить(ИдКолонки);
        КонецПопытки;
        Если Список.Пометка(Сч) = 1 Тогда
            Колонка.Видимость(1);
            Если ИдКолонки = ТекущаяКолонка Тогда
                Конт.Активизировать(ИдКолонки,0);
            КонецЕсли;
        Иначе
            Колонка.Видимость(0);
        КонецЕсли;
    КонецЦикла;
    СохранитьЗначение("СписокКолонок_" + ИдентификаторФормы, Список);
    СохранитьЗначение("ТекущаяКолонока_" + ИдентификаторФормы,
                      ТекущаяКолонка);
КонецПроцедуры // Принять()
```

В приведенной процедуре `ПриОткрытии()` присутствует метод `Форма.ИспользоватьЗакладки(1)`. В модуле формы нашей обработки `НастройкаКолонок` определим процедуру `ПриВыбореЗакладки()`:

```
Процедура ПриВыбореЗакладки(НомерСлоя,Слой)
    Форма.ИспользоватьСлой("Основной,"+Слой,2);
    Если Слой = "Активизировать"  Тогда
      Актив.УдалитьВсе();
      Для Сч = 1 По Видим.РазмерСписка() Цикл
          Если Видим.Пометка(Сч) = 1 Тогда
               Представление = "";
               Значение = Видим.ПолучитьЗначение(Сч,Представление);
               Актив.ДобавитьЗначение(Значение,Представление);
          КонецЕсли;
      КонецЦикла;
      Актив.ТекущаяСтрока(макс(Актив.НайтиЗначение
                                  (ВремТекКолонка),1));
    Иначе
      ВремТекКолонка = Актив.ПолучитьЗначение
                     (Макс (Актив.ТекущаяСтрока(),1));
    КонецЕсли;
КонецПроцедуры // ПриВыбореЗакладки()
```

Глобальный модуль

Глобальный модуль это, прежде всего, модуль основной стартовой формы программы. В Глобальном модуле определяются переменные, процедуры и функции, доступные для вызова из любых модулей форм и документов. Процедуры и функции, которые повторяются для одинаковых расчетов, в ряде модулей лучше определить один раз, настроив при этом соответствующим образом параметры и порядок выхода из них. Приведенный ранее вариант настройки колонок справочника "Квартиры" хотелось бы применить и к другим справочникам, не определяя в каждом модуле формы справочника процедуру `УстановитьВидимостьКолонок()`. Перенесем данную процедуру в глобальный модуль, предварительно изменив настройку параметров.

Внимание

Для лучшей читабельности текстов модулей, названия процедур и функций в глобальном модуле начинаются с префикса "гл", но это вовсе не означает синтаксического ограничения на названия процедур глобального модуля. На самом деле правила формирования названий процедур и функций в глобальном модуле такие же, как и в остальных модулях форм и документов.

```
// УПРАВЛЕНИЕ ВИДИМОСТЬЮ КОЛОНОК В СПРАВОЧНИКАХ
// Параметры:
// Конт - контекст формы справочника, из которого вызвана процедура.
// ИдентификаторФормы – уникальный идентификатор формы, из которой
//выполнен вызов, используется для сохранения пользовательских
// настроек.
// НачальныйСписок – список колонок, видимость которых подлежит
//настройке.
// ПриОткрытии – признак того, что вызов процедуры был сделан при
//открытии формы
Процедура глУстановитьВидимостьКолонок (Конт, ИдентификаторФормы,
          НачальныйСписок, ПриОткрытии=0)  Экспорт
   Перем СписокКолонок, ТекущаяКолонка, ИдКолонки, Колонка;
   Если ПриОткрытии=0 Тогда
     СписокПараметров= СоздатьОбъект("СписокЗначений");
     СписокПараметров.ДобавитьЗначение(Конт);
     СписокПараметров.ДобавитьЗначение(ИдентификаторФормы);
     СписокПараметров.ДобавитьЗначение(НачальныйСписок);
    ОткрытьФормуМодально("Обработка.НастройкаКолонок",
                         СписокПараметров);
   Иначе
     СписокКолонок = ВосстановитьЗначение("СписокКолонок_" +
                                          ИдентификаторФормы);
```

```
    ТекущаяКолонка=ВосстановитьЗначение ("ТекущаяКолонока_" +
                                        ИдентификаторФормы);
    Если ПустоеЗначение(СписокКолонок) = 1 Тогда
      НачальныйСписок.Выгрузить(СписокКолонок);
      СохранитьЗначение("СписокКолонок_" + ИдентификаторФормы,
                        СписокКолонок);
  ИначеЕсли ТипЗначенияСтр(СписокКолонок) = "СписокЗначений" Тогда
      Если СписокКолонок.РазмерСписка() <> НачальныйСписок.
           РазмерСписка() Тогда
        НачальныйСписок.Выгрузить(СписокКолонок);
        СохранитьЗначение("СписокКолонок_" +
                          ИдентификаторФормы, СписокКолонок);
      КонецЕсли;
    КонецЕсли;
    Для Сч = 1 По СписокКолонок.РазмерСписка() Цикл
      ИдКолонки   =      СписокКолонок.ПолучитьЗначение(Сч);
      Попытка
        Колонка  = Конт.Форма.ПолучитьАтрибут(ИдКолонки);
      Исключение
        Сообщить(ИдКолонки);
      КонецПопытки;
      Если СписокКолонок.Пометка(Сч) = 1 Тогда
        Колонка.Видимость(1);
        Если ИдКолонки = ТекущаяКолонка Тогда
          Конт.Активизировать(ИдКолонки,0);
        КонецЕсли;
      Иначе
        Колонка.Видимость(0);
      КонецЕсли;
    КонецЦикла;
  КонецЕсли;
КонецПроцедуры // глУстановитьВидимостьКолонок
```

Предопределенные процедуры Глобального модуля

Для стартовой формы программы, как и для любой другой формы, существуют предопределенные процедуры, которые определяются в Глобальном модуле:

- `ПриНачалеРаботыСистемы()`;
- `ПриЗавершенииРаботыСистемы()`.

Кроме того, в Глобальном модуле определяются предопределенные процедуры для форм, которые недоступны в Конфигураторе, т. е. формы истории и констант:

- ПриОткрытииИстории();
- ПриЗаписиИстории();
- ПриУдалении(Истории();
- ПриЗаписиКонстанты().

Ряд предопределенных процедур Глобального модуля предназначены для обработки интерактивных действий пользователя, влияющих на работу системы в целом:

- ПриУдалении(Элемента();
- ПриУдалении(Документа();
- ПриИзменении(ВремениДокумента();
- ПриОтмене(ПроведенияДокумента();
- ПриУстановкеОтбора();
- ПриУдалении(Счета();
- ПриВыклВклПроводокОперации();
- ПриИзменении(ВремениДокумента();
- ПриСмене(РасчетногоПериода().

1. ПриНачалеРаботыСистемы().

Предопределенная процедура выполняется при начале работы системы. Ее назначение — инициализация начальных значений некоторых справочников и констант при первом запуске программы; выполнение ряда действий при специфическом обновлении релиза программы; выполнение некоторых действий при анализе даты запуска программы, имени пользователя и т. п. Процедуру необходимо выполнять до начала работы пользователя с системой.

Рассмотрим анализ запуска программы на предмет первого запуска.

```
Процедура ПриНачалеРаботыСистемы()
   ЭтоПервыйЗапуск = 0;
   Если ПустоеЗначение(Константа.НомерРелиза) = 1 Тогда
      ЭтоПервыйЗапуск = 1;
      Док = СоздатьОбъект("Документ");
      Если Док.ВыбратьДокументы() = 1 Тогда
         ЭтоПервыйЗапуск = 0;
      КонецЕсли;
      Если ЭтоПервыйЗапуск = 1 Тогда
```

```
        ПоставляемыеСправочники = ",КлассификаторЕН,";
        Индекс = 1;
        КоличествоСправочников = Метаданные.Справочник();
Пока (ЭтоПервыйЗапуск=1) и (Индекс <= КоличествоСправочников) Цикл
            ВидСпр = Метаданные.Справочник(Индекс).Идентификатор;
       Если Найти(ПоставляемыеСправочники, ","+ВидСпр+",") = 0 Тогда
              Спр = СоздатьОбъект("Справочник."+ВидСпр);
              Если Спр.ВыбратьЭлементы(0) = 1 Тогда
                ЭтоПервыйЗапуск = 0;
              КонецЕсли;
            КонецЕсли;
            Индекс = Индекс+1;
        КонецЦикла;
    КонецЕсли;
    Если ЭтоПервыйЗапуск = 1 Тогда
        Константа.НомерРелиза = Лев(Метаданные.Комментарий, 8);
        ОткрытьФормуМодально("Обработка.ПервыйЗапуск", 1);
    Иначе
        Константа.НомерРелиза = "7.70.001";
    КонецЕсли;
КонецЕсли;
ПервыйЗапускНаРабочемМесте = ВосстановитьЗначение ("ПервыйЗапуск-
                              НаРабочемМесте");
Если ПустоеЗначение(ПервыйЗапускНаРабочемМесте) = 1 Тогда
    СохранитьЗначение("ОсновнаяВалюта", Константа.ОсновнаяВалюта);
    СохранитьЗначение("ОсновнойСклад", Константа.ОсновнойСклад);
    СохранитьЗначение("ОсновнойБанковскийСчет",
                      Константа.ОсновнойБанковскийСчет);
    СохранитьЗначение("ОсновноеПодразделение",
                     Константа.ОсновноеПодразделение);
    СохранитьЗначение("ОсновнаяСтавкаНДС",
                      Константа.ОсновнаяСтавкаНДС);
    СохранитьЗначение("ОсновнаяСтавкаНП",
                      Константа.ОсновнаяСтавкаНП);
    СохранитьЗначение("ОсновнаяЕдиницаИзмерения",
                      Константа.ОсновнаяЕдиницаИзмерения);
    СохранитьЗначение("ПоказПутеводителяПриЗапуске", 1);
    СохранитьЗначение("ПервыйЗапускНаРабочемМесте", 1);
    РазворотСубконто = СоздатьОбъект("ТаблицаЗначений");
```

```
    РазворотСубконто.НоваяКолонка("Счет",,,, "Счет", 16);
    РазворотСубконто.НоваяКолонка("Субсчета",,,, "Субсчета", 9);
    РазворотСубконто.НоваяКолонка("Субконто",,,, "Субконто", 9);
    РазворотСубконто.НоваяКолонка("Список");
    РазворотСубконто.НоваяСтрока();
    РазворотСубконто.Счет = СчетПоКоду(51);
    РазворотСубконто.Субсчета = "-";
    РазворотСубконто.Субконто = "Х";
    РазворотСубконто.Список = СоздатьОбъект("СписокЗначений");
    РазворотСубконто.Список.ДобавитьЗначение
                (ВидыСубконто.БанковскиеСчета);
    РазворотСубконто.Список.Пометка(1, 1);
    СохранитьЗначение("ОСВРазворотСубконто", РазворотСубконто);
    СохранитьЗначение("ОСВДанныеПоСубсчетам", 1);
    СохранитьЗначение("ОСВЗабалансовыеСчета", 1);
  КонецЕсли;
  Если Константа.НомерРелиза <> Лев(Метаданные.Комментарий, 8) Тогда
    ОткрытьФорму("Обработка.ОбновлениеИБ");
  КонецЕсли;
  ЗаголовокСистемы(СокрЛП(Константа.НазваниеОрганизации));
КонецПроцедуры // ПриНачалеРаботыСистемы()
```

2. `ПриЗавершенииРаботыСистемы()`.

Данная предопределенная процедура выполняется при завершении работы пользователя. Рассмотрим данную процедуру на предмет запроса на подтверждение окончания работы. Пользователь часто "промахивается", нажимая кнопку закрытия программы вместо кнопки закрытия текущего окна в полноэкранном режиме.

```
Процедура ПриЗавершенииРаботыСистемы()
  Перем ЗапрашиватьПодтверждениеПриВыходе;
  ЗапрашиватьПодтверждениеПриВыходе =
  ПустоеЗначение( ВосстановитьЗначение
  ("НеЗапрашиватьПодтверждениеПриВыходе"));
  Если ЗапрашиватьПодтверждениеПриВыходе = 1 Тогда
 Если Вопрос("Закончить работу с программой?", "Да+Нет", 60)="Нет"
Тогда
    СтатусВозврата(0);
    Возврат;
  КонецЕсли;
  КонецЕсли;
КонецПроцедуры  // ПриЗавершенииРаботыСистемы()
```

3. ПриУдаленииЭлемента(УдалЭлем,Режим).

Данная процедура выполняется при интерактивном удалении элемента справочника.

Параметры:

- УдалЭлем — удаляемый элемент;
- Режим — режим удаления:
  - 1 — непосредственное удаление,
  - 0 — пометка на удаление.

Данная предопределенная процедура практически не используется, т. к. механизм удаления элементов справочника, с отслеживанием подчинения и участия в качестве субконто в проводках, достаточно хорошо организован в программе. Однако бывают ситуации, когда необходимо организовать непосредственное удаление элементов. Рассмотрим простейший пример:

```
Процедура ПриУдаленииЭлемента(УдалЭлем,Режим)
  Если УдЭл.ПометкаУдаления()=0 Тогда
  Если Вопрос("Объект будет удален безвозвратно! Удалить?",1,30) = 1
тогда
        Сп=СоздатьОбъект("Справочник."+УдЭл.ПредставлениеВида());
        Сп.НайтиПоКоду(УдЭл.Код,0);
        Сп.Удалить(1);
     КонецЕсли;
   КонецЕсли;
КонецПроцедуры  // ПриУдаленииЭлемента
```

4. ПриОткрытииИстории(ТипОбъекта,Объект,ТолькоПросмотр).

Данная процедура выполняется при открытии окна истории значения периодического реквизита элемента справочника или периодической константы.

Параметры:

- ТипОбъекта — строка с названием периодического объекта конфигурации (периодического реквизита справочника или периодической константы);
- Объект — элемент справочника, для которого выполняется открытие окна истории периодического реквизита (для константы не используется);
- ТолькоПросмотр — флаг установки режима "только просмотр" для окна истории значения.

Если значение параметра ТолькоПросмотр установить в 1 (в предопределенной процедуре), то окно истории будет открыто только для чтения. Установка значения в 0 введет режим по умолчанию (определенный правами). Открытое окно истории соответствующим образом изменит режим (если это

случай, когда процедура обрабатывает не открытие окна, а смену отображаемого объекта). Значение по умолчанию `ТолькоПросмотр` — 0.

5. `ПриЗаписиИстории(ТипОбъекта,Объект,Значение,ДатаИстории)`.

Данная процедура выполняется при записи в окне истории значения периодического реквизита элемента справочника или периодической константы.

Параметры:

- `ТипОбъекта` — строка с названием периодического объекта конфигурации (периодического реквизита справочника или периодической константы);
- `Объект` — элемент справочника, у которого выполняется изменение значения периодического реквизита (для константы не используется);
- `Значение` — введенное значение;
- `ДатаИстории` — дата, на которую введено значение.

Рассмотрим пример контроля записи в историю, если пользователем не было задано конкретное значение. Пользователь должен подтвердить запись в историю нулевого значения константы `ПределСуммыДолга`:

```
Процедура ПриЗаписиИстории(ТипОбъекта, Объект, Значение,
                                                ДатаИстории)
    Если ТипОбъекта = "Константа.ПределСуммыДолга" Тогда
        Если ПустоеЗначение(Значение) = 1 Тогда
            Предупреждение("Сумма долга не может равняться 0!",30);
            СтатусВозврата(0);
        КонецЕсли;
    КонецЕсли;
КонецПроцедуры  // ПриЗаписиИстории()
```

6. `ПриУдаленииИстории(ТипОбъекта,Объект,Значение,ДатаИстории)`.

Данная процедура выполняется при удалении в окне истории значения периодического реквизита элемента справочника или периодической константы.

Параметры:

- `ТипОбъекта` — строка с названием периодического объекта конфигурации (периодического реквизита справочника или периодической константы);
- `Объект` — элемент справочника, у которого выполняется удаление значения периодического реквизита (для константы не используется);
- `Значение` — удаляемое значение;
- `ДатаИстории` — дата, на которую было записано значение.

7. `ПриЗаписиКонстанты(ИмяКонстанты,Значение)`.

Данная процедура выполняется при интерактивном вводе значения константы.

Параметры:

- ИмяКонстанты — строка с названием константы;
- Значение — введенное значение.

Процедура применяется аналогично процедуре ПриЗаписиИстории(), например:

```
Процедура ПриЗаписиКонстанты(ИмяКонстанты, Значение)
    Если ИмяКонстанты = "НомерРелиза" Тогда
   Предупреждение("Константа ""Номер релиза"" является служебной и
                  |не может быть изменена вручную.");
      СтатусВозврата(0);
КонецПроцедуры  //ПриЗаписиКонстанты
```

8. ПриУдаленииДокумента(УдалДокум,Режим).

Данная процедура выполняется при интерактивном удалении документа.

Параметры:

- УдалДокум — удаляемый документ;
- Режим — режим удаления:
  - 1 — непосредственное удаление;
  - 0 — пометка на удаление.

Используется ПриУдаленииДокумента() для организации запрета удаления документа:

```
Процедура ПриУдаленииДокумента(Докум, Режим)
    Если Докум.ДатаДок <= Константа.ДатаЗапретаРедактирования Тогда
      СтатусВозврата(0);
      Если Докум.ПометкаУдаления() = 0 Тогда
     Предупреждение("Нельзя удалять документы с датой, более ранней,
                    чем дата запрета редактирования документов!");
      Иначе
        Предупреждение("Нельзя отменять удаление документов с датой,
                       более ранней, чем дата запрета редактирования
                       документов!");
      КонецЕсли;
    КонецЕсли;
КонецПроцедуры
```

9. ПриИзмененииВремениДокумента(Докум).

Данная процедура выполняется при интерактивном изменении времени документа.

Специфическая процедура обычно применяется с компонентой **Оперативный учет**.

Параметр — Докум является обрабатываемым документом.

10. `ПриОтменеПроведенияДокумента(Докум)`.

Данная процедура выполняется при интерактивной отмене проведения документа.

Параметр — `Докум` является обрабатываемым документом.

Процедура `ПриУдалении Документа()` используется для организации запрета отмены проведения документа:

```
Процедура ПриОтменеПроведенияДокумента(Докум)
    Если Докум.ДатаДок <= Константа.ДатаЗапретаРедактирования Тогда
        Предупреждение("Нельзя отменять проведение документов с датой,
                        более ранней, чем дата запрета редактирования
                        документов!");
        СтатусВозврата(0);
        Возврат;
    КонецЕсли;
КонецПроцедуры
```

11. `ПриУстановкеОтбора(ИмяРеквизОтбора,Значение)`.

Данная процедура выполняется при интерактивной установке отбора документов в журнале.

Параметры:

- `ИмяРеквизОтбора` — строка с названием общего реквизита документа, по которому производится отбор;
- `Значение` — значение реквизита отбора.

Процедуру можно использовать для задания дополнительного условия отбора.

12. `ПриУдалении Счета(УдалСчет,Режим)`.

Данная процедура выполняется при удалении бухгалтерского счета.

Параметры:

- `УдалСчет` — удаляемый бухгалтерский счет (значение типа "Счет");
- `Режим` — режим удаления, может принимать значения:
  - `1` — счет будет удален;
  - `0` — счет будет помечен на удаление.

13. `ПриВыклВклПроводокОперации(Документ)`.

Данная процедура выполняется при выключении или включении бухгалтерских проводок.

Параметр `Документ` — значение типа "Документ" (документ), которому принадлежит операция.

Процедура ПриУдаленииДокумента() используется для организации запрета выключения и включения проводок:

```
Процедура ПриВыклВклПроводокОперации(Докум)
    Если Докум.ДатаДок <= Константа.ДатаЗапретаРедактирования Тогда
        СтатусВозврата(0);
        Если Докум.Операция.ВключитьПроводки() = 0 Тогда
            Предупреждение("Нельзя включить проводки документов
                с датой, более ранней, чем дата запрета
                редактирования документов!");
        Иначе
            Предупреждение("Нельзя выключить проводки документов
                с датой, более ранней, чем дата запрета
                редактирования документов!");
        КонецЕсли;
    КонецЕсли;
КонецПроцедуры
```

14. ПриСменеРасчетногоПериода(ЖурналРасчетов,Период).

Данная процедура выполняется при смене текущего расчетного периода журнала расчетов.

Параметры:

- ЖурналРасчетов — журнал расчетов, период которого изменяется (агрегатный объект типа "ЖурналРасчетов");
- Период — устанавливаемый расчетный период (значение типа "ПериодРасчета").

С помощью процедуры организуется запуск помощника для смены периода:

```
Процедура ПриСменеРасчетногоПериода(ЖурналРасчетов,Период)
  Если ВидЖурнала = "Начисления" Тогда
    Если ЖурналРасчетов.КонецТекущегоПериода()+1=Период.ДатаНачала
    Тогда
        ОткрытьФорму("Обработка.ПомощникПереходаНаСледующийПериод");
        СтатусВозврата(0);
    КонецЕсли;
  КонецЕсли;
КонецПроцедуры        // ПриСменеРасчетногоПериода
```

15. ОбработкаЯчейкиТаблицы(Значение,ФлагСтандОбраб,Таблица,Адрес).

Данная процедура обработки ячейки таблицы выполняется по двойному щелчку кнопки мыши или нажатием клавиши <Enter> в табличном документе на выбранной ячейке.

Параметры:

- `Значение` — здесь в процедуру передается вычисленное значение ячейки (в Конфигураторе задается: "Свойства ячейки", "Текст", "Значение");
- `ФлагСтандОбраб` — флаг обработки ячейки (установка в 1 приведет к выполнению стандартной обработки значения ячейки по завершении процедуры или открытию документа, элемента справочника и т. п.);
- `Таблица` — имя переменной (необязательный параметр), куда система передаст объект типа "Таблица";

Примечание

С помощью значения этого контекста можно произвольно манипулировать данной таблицей, пока она открыта. Пока объект "Таблица" существует, тип значения данного параметра равен `100`, если закрыт — `0`;

- `Адрес` — необязательный параметр. Имя переменной, куда система передаст адрес ячейки или объекта в формате "R1C1:R2C2".

Внимание

Объект типа "СписокЗначений" может записываться в поле "значение" ячейки таблицы и использоваться затем процедурой `ОбработкаЯчейкиТаблицы`. Если данная процедура описана в модуле формы, то вызывается именно она, иначе система запускает одноименную процедуру из глобального модуля. И наконец, данная предопределенная процедура не вызывается при выборе ячейки таблицы в режиме ввода данных. Для этого случая вызывается предопределенная процедура `ПриВыбореЯчейкиТаблицы`.

Приведем в качестве примера процедуру `ОбработкаЯчейкиТаблицы()` из типовой конфигурации как наиболее показательный вариант:

```
Процедура ОбработкаЯчейкиТаблицы(Расшифровка, СтандартнаяОбработка,
          Таблица) Экспорт
  Перем Отчет;
  Если ТипЗначенияСтр(Расшифровка) <> "СписокЗначений" Тогда
    СтандартнаяОбработка = 1;
    Возврат;
  КонецЕсли;
// Расшифровка отчета
  Док = Расшифровка.Получить("Документ");
  Если ТипЗначения(Док) <> 0 Тогда
    ОткрытьФорму("Операция",, Док, Расшифровка.Получить
                ("НомерПроводки"),Расшифровка.Получить
                        ("НомерКорреспонденции"),-1);
```

```
Иначе
    Меню = Расшифровка.Получить("Меню");
  Если ТипЗначенияСтр(Меню) = "СписокЗначений" Тогда
     Если Меню.ВыбратьЗначение(Отчет, "",,, 1) = 0 Тогда
        Возврат;
     КонецЕсли;
  Иначе
     Отчет = Расшифровка.Получить("Отчет");
  КонецЕсли;
  глРасшифровка = Расшифровка;
  глФлагРасшифровки = 1;
  глОбновить = Число(Расшифровка.Получить("Обновить"));
  Если глОбновить <> 0 Тогда
    глТаблица = Таблица;
  КонецЕсли;
  Если Отчет = "ОборотноСальдоваяВедомостьПоСчету" Тогда
     Счет = Расшифровка.Получить("Счет");
     Если Счет.КоличествоСубконто() = 0 Тогда
        Расшифровка.Установить("ДанныеПоСубсчетам",
                                        Счет.ЭтоГруппа());
     КонецЕсли;
  КонецЕсли;
  Если Расшифровка.Получить("ОткрытьЖурналДокументов") = 1 Тогда
     Док = Расшифровка.Получить("ДокументЖурнала");
     Конт = ПолучитьПустоеЗначение("");
     ОткрытьФорму("Журнал.Общий#ЖДОб", Конт);
     Конт.УстановитьИнтервал(Док.ДатаДок - 30, Док.ДатаДок + 30);
     Конт.АктивизироватьОбъект(Док);
  Иначе
     Попытка
        Если Метаданные.Отчет(Отчет).Выбран() = 1 Тогда
          ОткрытьФорму("Отчет."+Отчет+"#");
        ИначеЕсли Метаданные.Обработка(Отчет).Выбран() = 1 Тогда
          ОткрытьФорму("Обработка."+Отчет+"#");
        ИначеЕсли ФС.СуществуетФайл(Отчет) = 1 Тогда
          ОткрытьФорму("Отчет",, Отчет);
        КонецЕсли;
```

```
            Исключение
                Сообщить(ОписаниеОшибки());
            КонецПопытки;
        КонецЕсли;
        глФлагРасшифровки = 0;
        глРасшифровка = 0;
        глОбновить = 0;
    КонецЕсли;
КонецПроцедуры  // ОбработкаЯчейкиТаблицы
```

Навигация в теле модуля

Тело модуля формы или документа порой содержат множество процедур и функций со сложной вложенной структурой. Структура модуля представлена в специальном окне **Процедуры и функции модуля** (см. рис. 2.35).

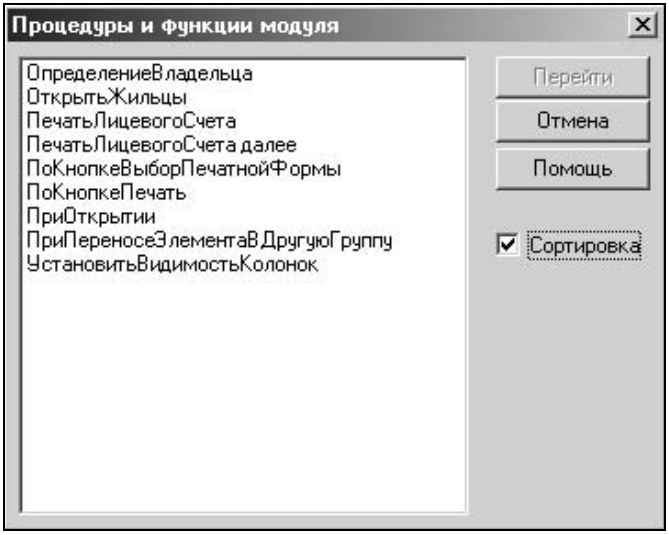


Рис. 2.35. Диалоговое окно выбора процедур и функций модуля формы списка справочника "Квартиры"

Окно вызывается из меню **Действия | Процедуры и функции модуля** в режиме редактирования модуля. По двойному щелчку кнопки мыши на выбранном пункте, указатель (курсор) становится на начало соответствующей процедуры или функции.

Синтакс-Помощник

Синтакс-Помощник — средство, облегчающее разработку модулей. Основная задача его — предоставить специалисту, выполняющему конфигурирование системы 1С:Предприятие, оперативную подсказку по встроенному языку.

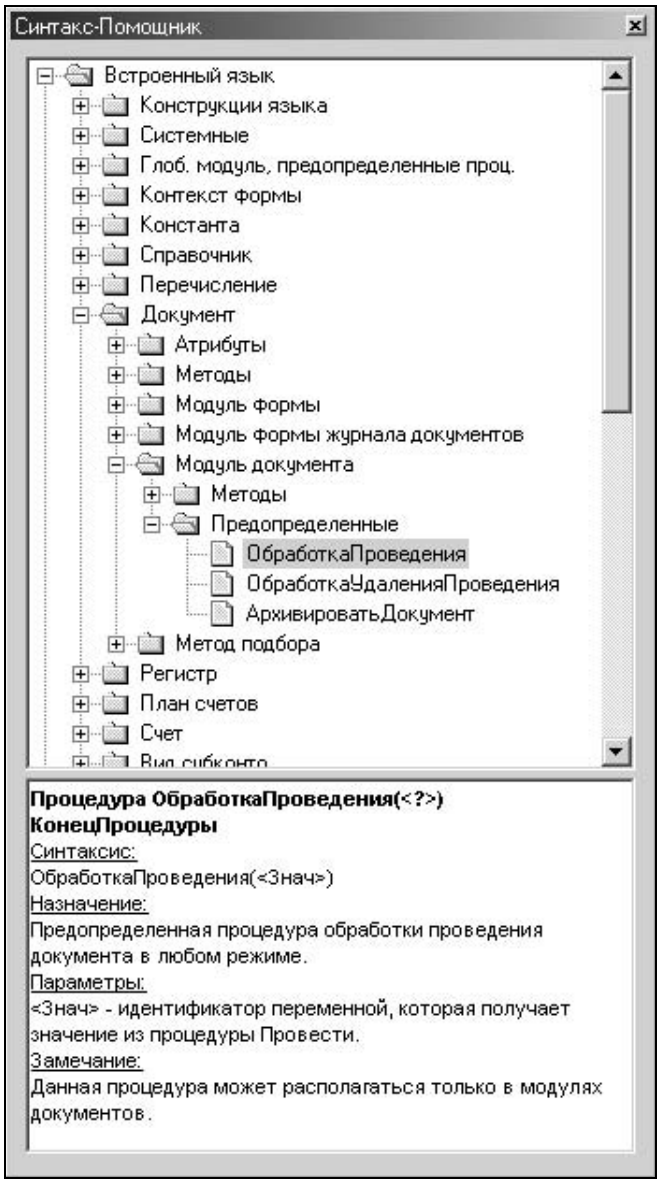


Рис. 2.36. Диалоговое окно Синтакс-Помощника

Для вызова Синтакс-Помощника выберите пункт **Сервис | Синтакс-Помощник** в меню Конфигуратора.

Окно Синтакс-Помощника состоит из двух частей. В верхней части, в виде дерева, выдается список элементов встроенного языка системы 1С:Предприятие: операторов, управляющих конструкций, процедур и функций, системных констант и др. Для удобства все элементы встроенного языка объединены в тематические разделы, представленные в виде ветвей дерева. Помимо элементов встроенного языка, дерево, в верхней части окна Синтакс-Помощника, содержит список существующих шаблонов. В нижней части окна Синтакс-Помощника выдается краткое описание элемента встроенного языка, выбранного в верхнем окне. Для получения описания следует дважды щелкнуть кнопкой мыши на наименовании элемента языка в верхней части окна. Для нахождения описания предопределенных процедур модуля документа необходимо раскрыть пункт **Встроенный язык**, затем раскрыть пункт **Документ** и после этого раскрыть пункт **Модуль документа**. Далее необходимо раскрыть пункт **Предопределенные** (см. рис. 2.36).

Для получения описания необходимо дважды щелкнуть мышью на выбранной процедуре.

Глава 3

Атрибуты и методы метаданных

При работе с объектами метаданных наибольший интерес представляют методы, реализующие действия, связанные с созданием новых элементов справочников, новых документов определенного вида; формированием выборок из справочников и документов; поиском конкретных записей, их удалением, сортировкой и т. п. По функциональной нагрузке эти методы аналогичны для различных типов метаданных.

Атрибуты метаданных

Атрибуты метаданных в общем случае задаются пользователем в Конфигураторе, кроме базовых полей, которые заданы непосредственно при их определении в исполняющем модуле программы.

Атрибуты справочников

Для справочников определены следующие атрибуты:

- Код;
- Наименование;
- Родитель;
- Владелец;
- Реквизит.

В приведенном выше перечне атрибуты — Код, Наименование, Родитель и Владелец являются базовыми. Реквизит — обобщенное название атрибутов справочника, которые задаются пользователем в Конфигураторе, в группе **Реквизиты**. Зададим в нашей конфигурации справочник "Контрагенты", который необходим для взаиморасчетов с клиентами, поставщиками и арендаторами. В окне настройки этого справочника, а именно в группе **Реквизиты**, будут перечислены все атрибуты, кроме базовых. Ими, в нашем случае, являются (см. рис. 3.1):

- **ВидКонтрагента**;

- **ПолнНаименование**;
- **ЮридическийАдрес**;
- **ПочтовыйАдрес**;
- **Телефоны**;
- **ИНН**;
- **ДокументСерия**;
- **ДокументНомер**;
- **ДокументКемВыдан**;
- **ДокументДатаВыдачи**;
- **ОсновнойДоговор**;
- **ОсновнойСчет**.

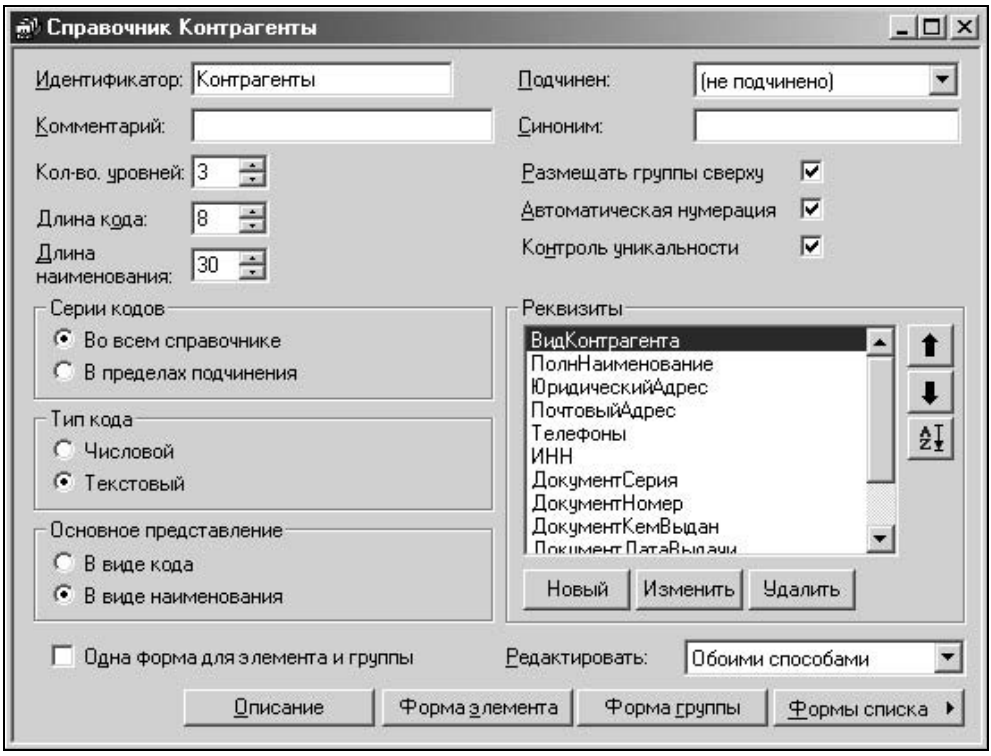


Рис. 3.1. Диалоговое окно справочника "Контрагенты"

Доступ к значениям атрибутов производится непосредственно по имени атрибута в элементе, на который спозиционирован указатель. Вернемся

к функции `ОпределениеВладельца()` в модуле формы списка справочника "Квартиры":

```
Функция ОпределениеВладельца()
   Жильцы = СоздатьОбъект("Справочник.Жильцы");
   Жильцы.ИспользоватьВладельца(ТекущийЭлемент());
   Жильцы.ВыбратьЭлементы();
   Пока Жильцы.ПолучитьЭлемент()=1 цикл
// Если указатель спозиционирован на искомого владельца квартиры
      Если Жильцы.СтатусПрописки = Перечисление.Статусы.Владелец
тогда
     // Возврат значения атрибута ФИО
            Возврат Жильцы.ФИО
      КонецЕсли;
   КонецЦикла;
КонецФункции
```

Существует специфический метод получения значения атрибута `Получить-Атрибут()`, который будет рассмотрен далее.

Атрибуты документов

Для документов определены следующие атрибуты:

- НомерДок;
- ДатаДок;
- НомерСтроки;
- Операция;
- Реквизит.

В приведенном перечне атрибуты — НомерДок, ДатаДок, НомерСтроки и Операция являются базовыми атрибутами. Реквизит — обобщенное название атрибутов Документа, которые задаются пользователем в конфигураторе в группах **Реквизиты шапки** и **Реквизиты табличной части**.

Следует обратить особое внимание на базовый атрибут документа Операция. Данный атрибут имеет тип `Документ.Операция`. С помощью этого атрибута осуществляется доступ к бухгалтерской операции документа. Атрибуты же самой операции будут рассмотрены в данной главе, в разделе — "Атрибуты операции".

Зададим в нашей конфигурации документ "УслугиСторонниxОрганизаций", который необходим для взаиморасчетов с поставщиками. В группе **Реквизиты шапки** перечислены все атрибуты шапки (см. рис. 3.2):

- **Контрагент**;
- **Договор**;

- **Валюта**;
- **ЗачитыватьАванс**;
- **ДатаНомерСчетаФактуры**;
- **ДокументПоступления**;
- **ВариантРасчетаНалогов**;
- **ТипИсполнителя**;
- **НДСвключатьВСтоимость**;
- **Курс**;
- **ВерсияОбъекта**;
- **НомерДокВходящий**;
- **ДатаДокВходящий**.

Доступ к значениям атрибутов шапки производится непосредственно по имени атрибута в документе, на который установлен курсор.

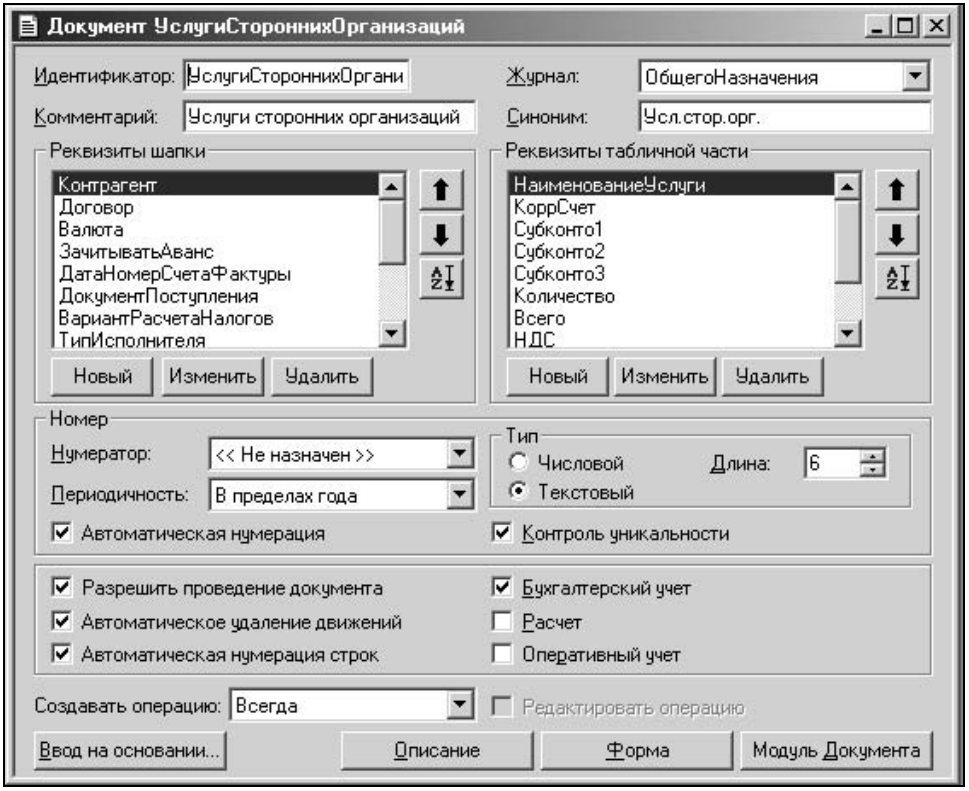


Рис. 3.2. Диалоговое окно документа "УслугиСторонниховОрганизаций"

В группе **Реквизиты табличной части**, как показано на рис. 3.2, перечислены все атрибуты табличной части документа:

- **НаименованиеУслуги**;
- **КоррСчет**;
- **Субконто1**;
- **Субконто2**;
- **Субконто3**;
- **Количество**;
- **Всего**;
- **НДС**;
- **НП**;
- **Сумма**.

Доступ к значениям атрибутов табличной части производится непосредственно по имени атрибута в строке документа, на которую установлен курсор.

> **Внимание**
>
> Для получения значения атрибута табличной части документа, необходимо установить курсор на нужный документ, а затем на строку табличной части документа.

Рассмотрим для примера фрагмент модуля формы документа "ФормированиеЗаписейКнигиПокупок", в котором из документов оприходования, подлежащих регистрации в книге покупок, выбираются только те, для которых в договорах включен признак АвтоОбработкаНДС:

```
Если (Док.Вид() = "УслугиСторонниxОрганизаций") или
     (Док.Вид() = "ПоступлениеТоваров") или
     (Док.Вид() = "ПоступлениеМатериалов") или
     (Док.Вид() = "ПоступлениеОборудования") или
     (Док.Вид() = "ПоступлениеОС") или
     (Док.Вид() = "СчетФактураПолученный") или
     (Док.Вид() = "ПоступлениеНМА") Тогда
// Обращение к атрибуту документа
// "УслугиСторонниxОрганизаций" (Реквизит Договор),
// затем проверка значения атрибута справочника
// "Договоры"- АвтоОбработкаНДС
     Если Док.Договор.АвтоОбработкаНДС=0 Тогда
          Продолжить;
     КонецЕсли;
```

```
    Если ПустоеЗначение(Док.ДатаНомерСчетаФактуры)=1 Тогда
        Продолжить;
    КонецЕсли;
КонецЕсли;
```

Атрибуты операций и проводок

Для операций и проводок определены следующие атрибуты:

- ДатаОперации;
- Содержание;
- СуммаОперации;
- Документ;
- Сумма;
- Валюта;
- ВалСумма;
- Количество;
- Дебет:
  - Счет;
  - Субконто;
- Кредит:
  - Счет;
  - Субконто;
- РеквизитОперации;
- РеквизитПроводки.

В приведенном перечне атрибуты — ДатаОперации, Содержание, СуммаОперации, Документ, Сумма, Валюта, ВалСумма, Количество, Дебет.Счет, Дебет.Субконто, Кредит.Счет, Кредит.Субконто являются базовыми. РеквизитОперации — обобщенное название атрибутов операции, которые задаются пользователем в Конфигураторе в группе **Реквизиты** (см. рис. 3.3). Подобно шапке документа, есть атрибуты, единые для всей операции. К ним относятся — ДатаОперации, Содержание, СуммаОперации и Документ. Доступ к их значениям производится непосредственно по имени атрибута в операции, на которую установлен курсор (указатель).

Следует обратить особое внимание на базовый атрибут операции — Документ, который содержит ссылку на документ, к которому относится данная операция. С помощью этого атрибута осуществляется доступ к исходному документу. Если он не задан, операция была создана вручную.

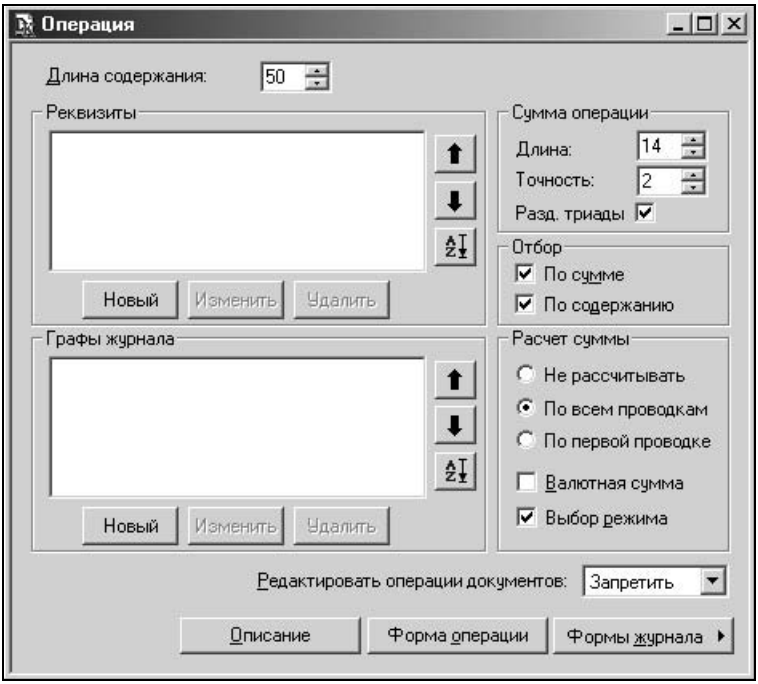


Рис. 3.3. Диалоговое окно **Операция**

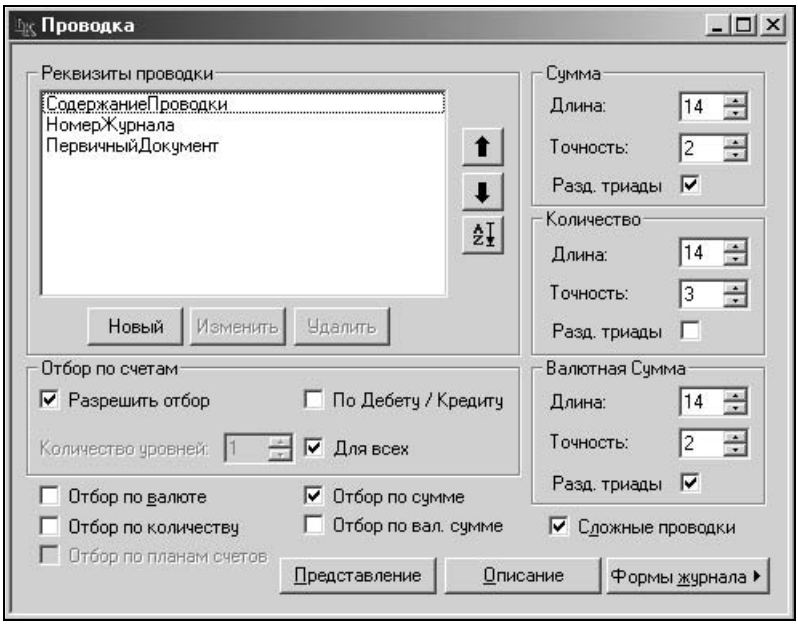


Рис. 3.4. Диалоговое окно **Проводка**

Подобно табличной части документа, доступ к значениям таких атрибутов проводки как — Сумма, Валюта, ВалСумма, Количество, Дебет.Счет, Дебет.Субконто, Кредит.Счет, Кредит.Субконто производится непосредственно по имени атрибута проводки, на которую установлен (спозиционирован) курсор.

РеквизитПроводки — обобщенное название атрибутов проводки, которые задаются пользователем в Конфигураторе в группе **Реквизиты проводки** (см. рис. 3.4).

Внимание

Доступ к проводкам невозможен отдельно от операции. Вначале указатель устанавливается на операцию, затем на нужную проводку.

Задание значений атрибутов операции применяется в модулях документов в предопределенной процедуре `ОбработкаПроведения()`. Рассмотрим модуль проведения нашего документа "Показания водомеров":

```
Процедура ОбработкаПроведения()
   ЖурналРасчетовНачислений =
   СоздатьОбъект("ЖурналРасчетов.Начисления");
   ДатаН = НачМесяца(ДатаДок);
   ДатаК = КонМесяца(ДатаДок);
   Пока ПолучитьСтроку()=1 Цикл
//Создадим записи в журнале расчетов
      Если ЖурналРасчетовНачислений.ВыбратьПериодПоОбъекту (Квартира,
           ЖурналРасчетовНачислений.КонецТекущегоПериода()) = 1 Тогда
         ЖурналРасчетовНачислений.УстановитьРеквизит
         ("НомерСтрокиДокумента",НомерСтроки);
         ЖурналРасчетовНачислений.ВвестиРасчет  (Квартира,Услуга.Расчет,
         ДатаН,ДатаК);
      Иначе
         ЖурналРасчетовНачислений.Новая();
         ЖурналРасчетовНачислений.УстановитьРеквизит
         ("Объект",Квартира);
         ЖурналРасчетовНачислений.УстановитьРеквизит
         ("ВидРасч",Услуга.Расчет);
         ЖурналРасчетовНачислений.УстановитьРеквизит
         ("Документ", ТекущийДокумент());
         ЖурналРасчетовНачислений.УстановитьРеквизит
         ("РодительскийДокумент",ТекущийДокумент());
         ЖурналРасчетовНачислений.УстановитьРеквизит
         ("ДатаНачала",ДатаН);
```

```
        ЖурналРасчетовНачислений.УстановитьРеквизит
        ("ДатаОкончания",ДатаК);
        ЖурналРасчетовНачислений.УстановитьРеквизит
        ("НомерСтрокиДокумента",НомерСтроки);
        ЖурналРасчетовНачислений.Записать();
        ЖурналРасчетовНачислений.Рассчитать();
    КонецЕсли;
КонецЦикла;
 // Создадим проводки
Операция.НоваяПроводка();
Операция.Дебет.Счет = Услуга.СчетДебета;
Операция.Дебет.Субконто(1,Услуга.СубконтоДебет1);
Операция.Дебет.Субконто(2,Услуга.СубконтоДебет2);
Операция.Дебет.Субконто(3,Услуга.СубконтоДебет3);
Операция.Кредит.Счет = Услуга.СчетКредита;
                    Операция.Кредит.Субконто
                    (1,Услуга.СубконтоКредит1);
Операция.Кредит.Субконто(2,Услуга.СубконтоКредит2);
Операция.Кредит.Субконто(3,Услуга.СубконтоКредит3);
Операция.Сумма = Итог("Сумма");
Операция.СодержаниеПроводки = "за хол. воду по водомеру";
Операция.Записать();
КонецПроцедуры
```

Значения базовых атрибутов ДатаОперации, СуммаОперации, Документ присваиваются автоматически при завершении обработки проведения.

Атрибуты регистров

Объекты метаданных типа Регистр используются компонентой Оперативный учет для накопления данных и отслеживания движений по заданному объекту. Для регистров определены следующие атрибуты:

- Приход;
- Расход;
- Измерение;
- Ресурс;
- Реквизит.

В приведенном перечне атрибуты — Приход и Расход являются базовыми. Атрибут — Измерение определяет обобщенное название атрибутов регистра, которые задаются пользователем в конфигураторе в группе **Измерение**.

Атрибут — Ресурс определяет обобщенное название атрибутов регистра, которые задаются пользователем в конфигураторе в группе **Ресурс**.

Атрибут — Реквизит определяет обобщенное название атрибутов регистра, которые задаются пользователем в конфигураторе в группе **Реквизиты**.

Зададим в нашей конфигурации регистр "Заявки", который необходим для учета заявок от жильцов на ремонт. В группе **Измерения**, как показано на рис. 3.5, перечислены следующие атрибуты учетных позиций:

- **Услуга**;
- **Квартира**;
- **ДатаИсполнения**;
- **Заявка**.

В группе **Ресурсы** перечислены все количественные и суммовые атрибуты (показатели) учетных позиций (рис. 3.5):

- **Количество**;
- **Долг**.

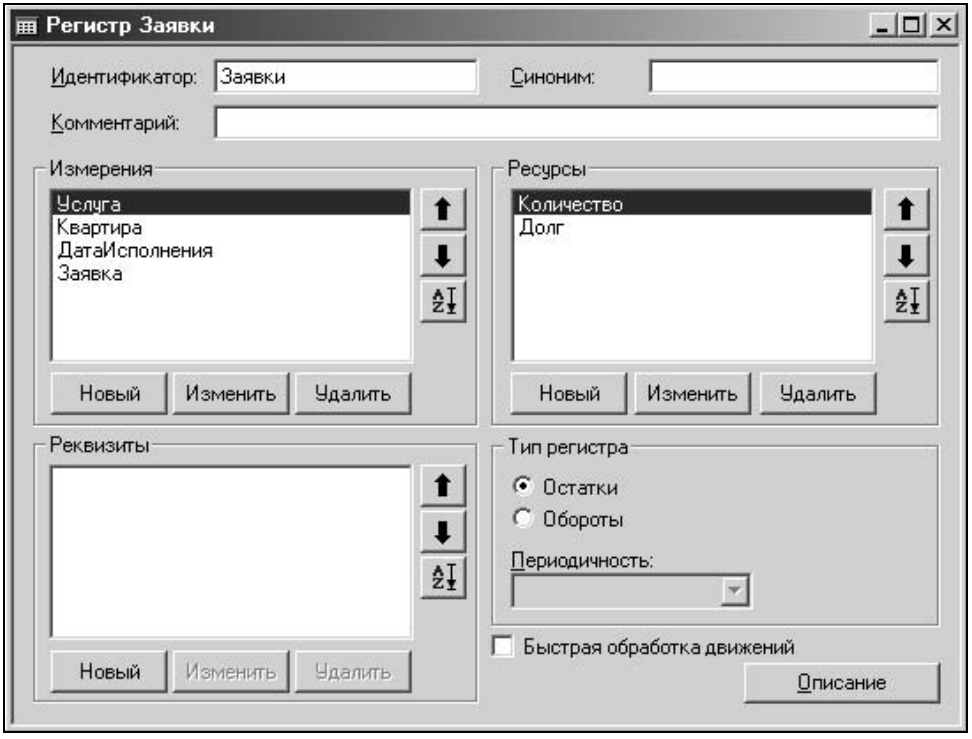


Рис. 3.5. Диалоговое окно регистра "Заявки"

В группе **Реквизиты** перечисляются все реквизиты, по которым необходим отбор (на рис. 3.5 они не указаны).

Задание значений атрибутов регистров обычно применяется в модулях документов в предопределенной процедуре `ОбработкаПроведения()`. Создадим **Документ Заявка** для ввода заявок на ремонт (рис. 3.6).

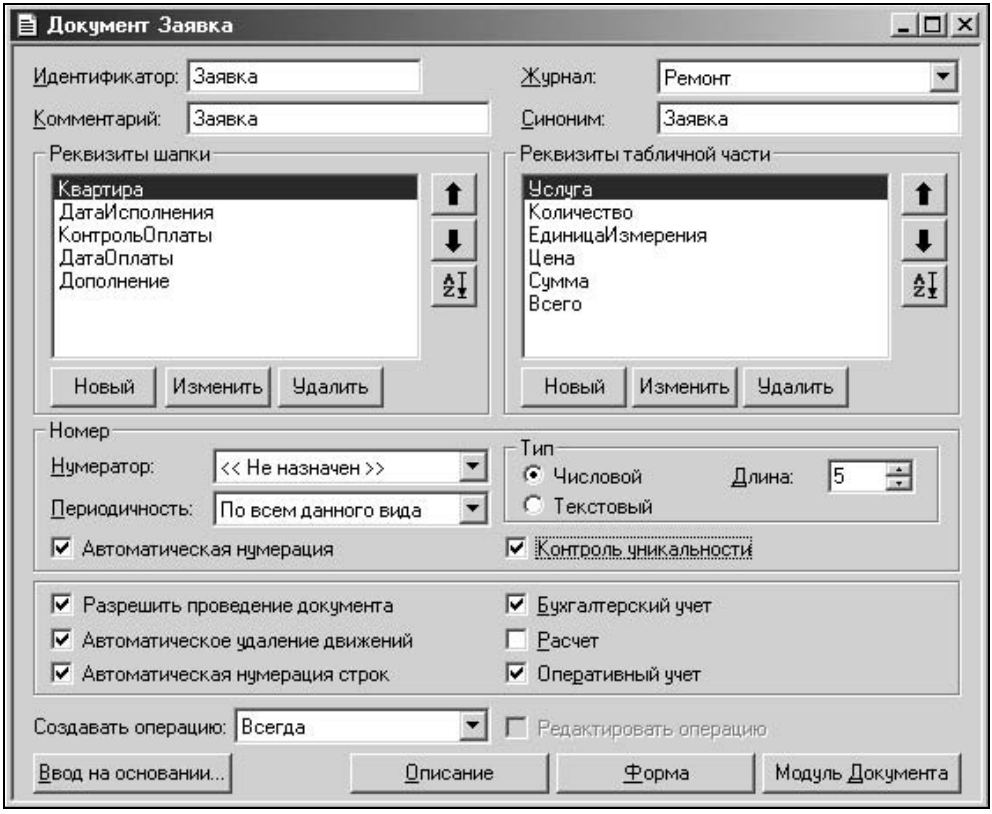


Рис. 3.6. Диалоговое окно **Документ Заявка**

Рассмотрим модуль проведения документа — Заявка:

```
Процедура ОбработкаПроведения()
    Регистр.Заявки.Квартира = Квартира;
    Регистр.Заявки.ДатаИсполнения = ДатаИсполнения;
    Регистр.Заявки.Заявка = ТекущийДокумент();
    Регистр.Заявки.Долг = 0;
    ВыбратьСтроки();
    Пока ПолучитьСтроку() = 1 Цикл
```

```
        Если ПустоеЗначение(Услуга) = 1 Тогда
          Сообщить("Строка:" + СокрЛП(НомерСтроки) +
                   ", не указан товар. Строка НЕ обработана!");
        Иначе
         // движение количества по регистру заявок
          Регистр.Заявки.Услуга = Услуга;
          Регистр.Заявки.Количество = Количество;
          Регистр.Заявки.ДвижениеПриходВыполнить();
      КонецЕсли;
    КонецЦикла;
    // движение сумм по регистру заявок
    Если КонтрольОплаты = 1 Тогда
      Регистр.Заявки.ДатаИсполнения = ДатаОплаты;
      Регистр.Заявки.Услуга = "";
      Регистр.Заявки.Количество = 0;
      Регистр.Заявки.Долг = Итог("Всего");
      Регистр.Заявки.ДвижениеПриходВыполнить();
    КонецЕсли;
КонецПроцедуры
```

Атрибуты видов расчетов

Для видов расчетов определены следующие атрибуты:

- Код;
- Наименование;
- Очередность;
- ПриоритетВытеснения.

Все атрибуты — базовые. Атрибут Наименование имеет тип "строка" и соответствует полю **Комментарий**, заданному при конфигурировании вида расчета. Атрибут — ПриоритетВытеснения имеет тип число. Он соответствует полю — **Приоритет**, заданному при конфигурировании вида расчета (рис. 3.7).

При работе с видами расчетов важная роль отведена правилам перерасчета.

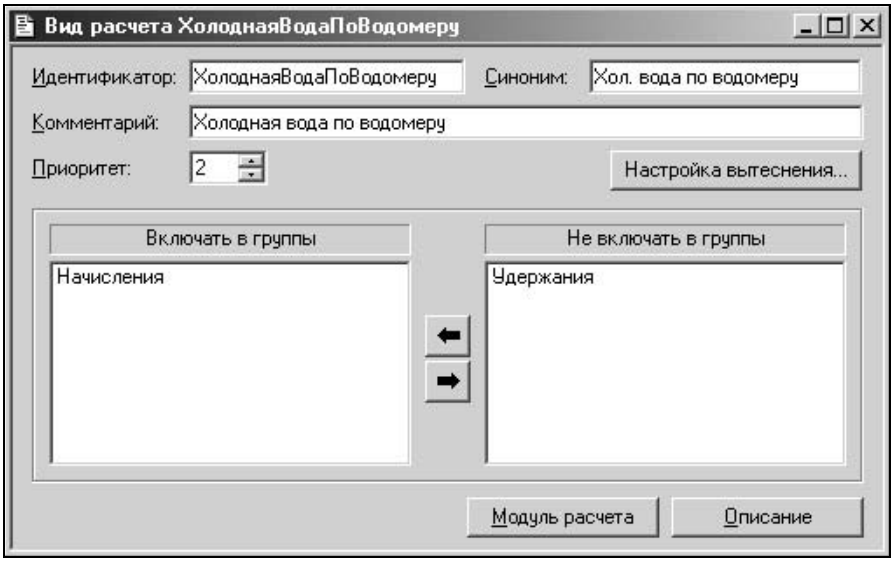


Рис. 3.7. Диалоговое окно **Вид расчета ХолоднаяВодаПоВодомеру**

Атрибуты правил перерасчета

Правила перерасчета задают алгоритм перерасчета зависимых видов расчета. Перерасчет одного вида расчета влечет перерасчет ряда других видов.

Для правил перерасчета определены следующие атрибуты:

- Тип;
- КоличествоПериодов.

Базовый атрибут Тип позволяет прочитать или установить значение типа правила перерасчета. Данный атрибут задается при конфигурировании переключателем и может принимать следующие значения (рис. 3.8):

- 0 — при вводе записи журнала расчетов с любым периодом действия зависимые виды расчетов должны быть перерасчитаны только в текущем периоде журнала расчетов (фактически это приведет к снятию флажка "рассчитанности" записи);
- 1 — зависимые виды расчетов должны быть перерасчитаны в том же периоде, что и вводимая запись журнала расчетов;
- 2 — зависимые виды расчетов должны быть перерасчитаны в нескольких периодах, следующих за периодом действия вводимой записи журнала расчетов (количество периодов задано атрибутом — КоличествоПериодов).

Базовый атрибут КоличествоПериодов позволяет прочитать или установить количество периодов перерасчета. Значение атрибута задается в поле при положении переключателя в нижней позиции. В диалоговом окне **Правило**

перерасчета переключатель, задающий значение атрибута Тип, расположен в левом нижнем углу окна (см. рис. 3.8).

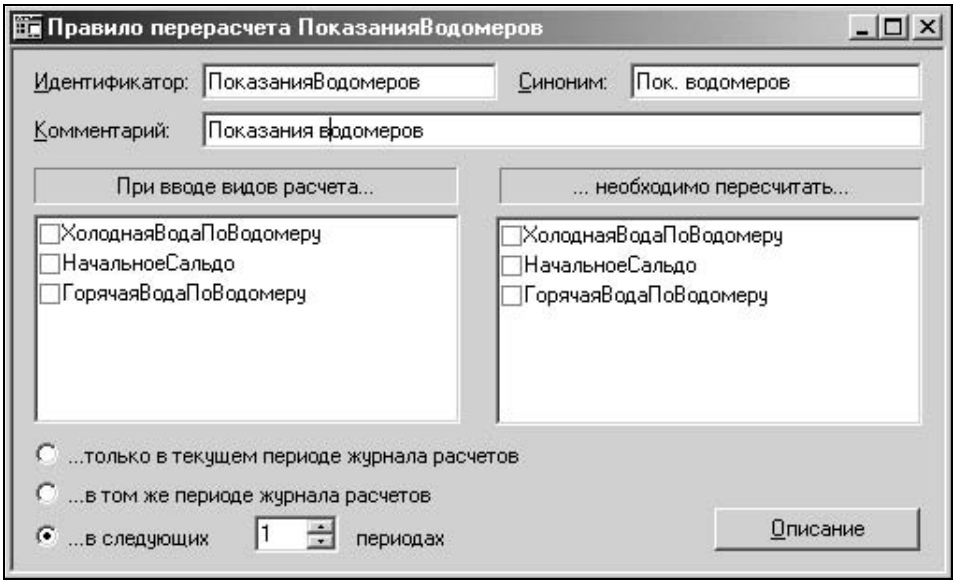


Рис. 3.8. Диалоговое окно перерасчета **ХолоднаяВодаПоВодомеру**

Атрибуты календаря

Для календарей определены следующие атрибуты:

- Дата;
- Значение.

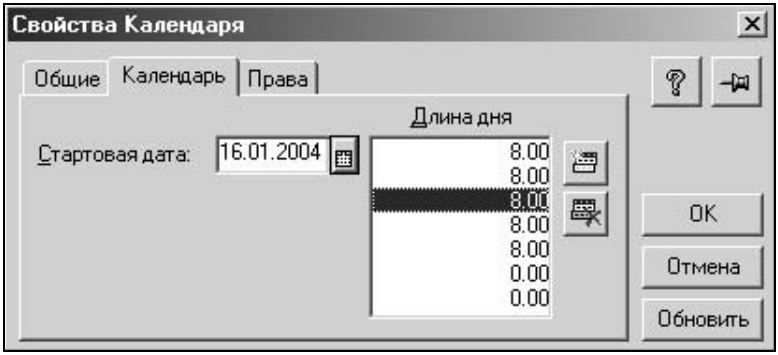


Рис. 3.9. Диалоговое окно свойств календаря

Атрибут — Дата, предназначенный только для чтения, соответствует дате, определенной в текущей строке календаря. Атрибут — Значение соответствует значению текущей (на которой позиционирована выборка) строки календаря. Атрибут доступен как для чтения, так и для записи. В диалоговом окне **Свойства Календаря** атрибуту Дата соответствует поле Стартовая дата, атрибуту Значение — поле Длина дня (рис. 3.9).

Атрибуты счетов

Здесь определены следующие атрибуты:

- Код;
- Наименование;
- Валютный;
- Количественный;
- Забалансовый;
- Активный;
- Реквизит.

В приведенном выше списке атрибуты — **Код**, **Наименование**, **Валютный**, **Количественный**, **Забалансовый** и **Владелец** являются базовыми (см. рис. 3.10).

План счетов Основной

Идентификатор: Основной Синоним: Основной план

Комментарий: Основной

Шаблон кода: ###.##.#

Код	Наименование	Вал.	Кол.	Заб.	Акт.	Субконто1	Субконто2
00	Вспомогательный				АП		
01	Основные средства				А	ОсновныеСредства	
01.1	ОС в организации				А	ОсновныеСредства	
01.2	Выбытие ОС				А	ОсновныеСредства	
02	Амортизация ОС				П	ОсновныеСредства	
02.1	Аморт. ОС, уч. на сч.01.1				П	ОсновныеСредства	
02.2	Аморт. ОС, уч. на сч.03				П	ОсновныеСредства	
03	Доходные вложения в МЦ				А	ОсновныеСредства	
03.1	МЦ в организации				А	ОсновныеСредства	
03.2	МЦ, перед. во врем.влад.				А	ОсновныеСредства	Контрагенты
03.3	МЦ, перед. во врем.польз.				А	ОсновныеСредства	Контрагенты
03.4	Прочие доходные вложения				А	ОсновныеСредства	Контрагенты
03.5	Выбытие МЦ				А	ОсновныеСредства	
04	Нематериальные активы				А	НематериальныеАкти	
04.1	Нематериальные активы				А	НематериальныеАкти	
04.2	Расходы на НИОКР				А	НематериальныеАкти	
05	Амортизация НМА				П	НематериальныеАкти	
07	Оборудование к установке		+		А	Оборудование	МестаХранения
08	Влож.во внеоборотн.активы				А		
08.1	Приобретение зем.участков				А	ОбъектыСтроительств	

Рис. 3.10. Диалоговое окно плана счетов

При помощи атрибута **Код** можно получать и задавать код счета. Код счета в общем случае представляет собой символьную строку вида:

```
Код счета.Код субсчета.Код субсчета ...
```

Общая длина кода счета ограничена 255 символами. При помощи атрибута **Наименование** можно получать и задавать наименование счета, которое представляет собой произвольную строку символов. Атрибут **Валютный** содержит признак ведения валютного учета по счету (1 или + в окне плана счетов валютный учет ведется по данному счету; 0 — валютный учет не ведется по данному счету). Атрибут **Количественный** содержит признак ведения количественного учета по счету (1 или + в окне плана счетов — количественный учет ведется по данному счету; 0 — количественный учет не ведется по данному счету). Атрибут **Забалансовый** содержит признак того, что счет является забалансовым. Забалансовый счет не участвует в двойной записи, не требует в проводках наличия корреспонденции и не может корреспондировать с балансовыми счетами (1 — выбранный счет является забалансовым счетом; 0 — выбранный счет является балансовым счетом). Атрибут **Активный** содержит тип остатка счета:

- 1 — счет является активным (в окне плана счетов — А);
- 2 — счет является пассивным (в окне плана счетов — П);
- 3 — счет является активно-пассивным (в окне плана счетов — АП);

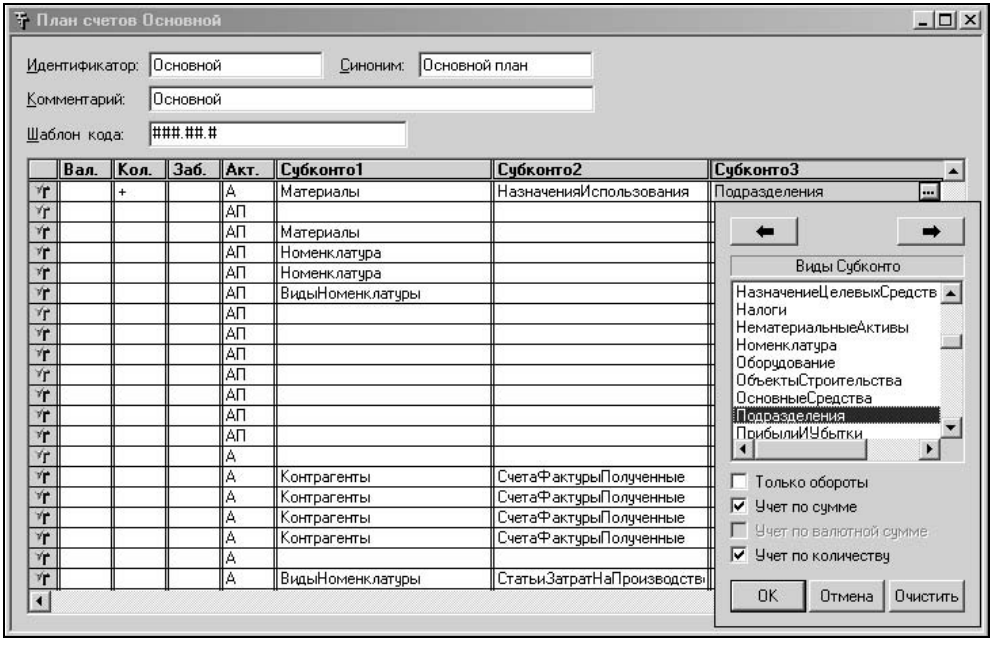


Рис. 3.11. Диалоговое окно **План счетов Основной** с расшифровкой **Субконто3**

Остатки активного счета должны быть дебетовыми, остатки пассивного должны быть кредитовыми, остатки активно-пассивного счета могут быть дебетовыми и кредитовыми. По умолчанию все счета считаются активно-пассивными.

Реквизит — обобщенное название атрибутов счета, которые задаются пользователем в Конфигураторе в графах **Субконто1**, **Субконто2**, **Субконто3** и т. д., в зависимости от заданного количества уровней аналитического учета (рис. 3.11).

Атрибуты журнала расчетов

Для журнала расчетов определены следующие атрибуты:

- **Документ** — документ, на основании которого была создана текущая запись журнала (атрибут только для чтения);
- **РодительскийДокумент** — документ, который ввел текущую запись в журнал расчетов (атрибут только для чтения);
- **Объект** — элемент справочника, для которого введена текущая запись журнала расчетов (атрибут только для чтения);
- **ВидРасч** — вид расчета текущей записи журнала расчетов (атрибут только для чтения);
- **ДатаНачала** — дата начала действия записи журнала расчетов (атрибут только для чтения);
- **ДатаОкончания** — дата окончания действия записи журнала расчетов (атрибут только для чтения);
- **ПериодДействия** — период действия записи журнала расчетов (атрибут только для чтения);
- **ПериодРегистрации** — период регистрации записи журнала расчетов (атрибут только для чтения);
- **Рассчитана** — признак того, что запись рассчитана (атрибут только для чтения), при этом признак может принимать значения:
  - 1 — для рассчитанных записей журнала расчетов;
  - 0 — для нерассчитанных записей;
- **Исправлена** — признак того, что запись исправлена вручную (атрибут только для чтения), при этом признак может принимать значения:
  - 1 — для записей журнала расчетов, результат которых исправлен вручную;
  - 0 для остальных записей;

- **Фиксирована** — признак того, что результат расчета записи защищен от исправления (атрибут только для чтения), при этом признак может принимать значения:
  - 1 — для фиксированных записей журнала расчетов;
  - 0 — для остальных записей;
- **Перерасчет** — признак того, что запись является перерасчетом другой записи прошлого периода (атрибут только для чтения), при этом признак может принимать значения:
  - 1 — для записей-перерасчетов;
  - 0 — для остальных записей;
- **ПервичнаяЗапись** — первичная запись записи-перерасчета (атрибут, только для чтения, типа — "Запись журнала расчетов");
- **Результат** — атрибут типа "число", который используется для доступа к результату расчета записи (как правило, самым важным действием процедуры модуля расчета — `ПровестиРасчет()` является вычисление результата расчета и заполнение данного атрибута);
- **Сторно** — признак сторнирующей записи, который может принимать значения:
  - 1 — для сторнирующих записей журнала расчетов;
  - 0 — для обычных записей.

Примечание

Признак — Сторно равен 1 не только для простых сторно-записей, но и для рассчитанных, отредактированных вручную или зафиксированных (не подлежащих редактированию) сторно-записей.

Если сторнирующая запись введена программным образом, т. е. атрибут `Сторно` задан за счет применения метода `УстановитьРеквизит()` или непосредственным присвоением:

```
ЖрнРасчета = СоздатьОбъект("ЖурналРасчета.Зарплата");
...
ЖрнРасчета.Сторно = 1;
...
```

тогда он может быть переопределен программным образом.

Общие методы объектов метаданных

Для работы с объектами метаданных наибольший интерес представляют методы задания новых объектов, получения и задания значений атрибутов, удаление объектов метаданных, а также поиск, позиционирование и выборка объектов с заданными признаками.

Рассмотрим методы, у которых совпадает функциональная нагрузка, названия и параметры.

1. `НазначитьТип()`.

Для установки типа не базовых атрибутов группы Реквизит, у которых в Конфигураторе задан тип — "Неопределенный" (**<<Неопределенный>>**), предназначен метод `НазначитьТип(ИмяРеквизита,ИмяТипа,Длина,Точность)` (см. рис. 3.12). Для объектов метаданных типа "Справочник", "Документ", "ЖурналРасчетов", "Константа", "Операции", "Регистр", "Счет" в методе определены следующие параметры:

- `ИмяРеквизита` — строка с наименованием реквизита неопределенного типа, как он назван в конфигураторе;
- `ИмяТипа` — строковое выражение, определяющее название типа данных (или Вид Субконто), который назначается реквизиту (например, "Строка", "Число", "Справочник.Товары", "Документ.РасходнаяНакладная" и т. п.);
- `Длина` — необязательный параметр типа "Число", определяющий длину поля представления данных (имеет смысл только при задании числового или строкового типа);
- `Точность` — необязательный параметр, определяющий число знаков после десятичной точки (имеет смысл только при задании числового типа).

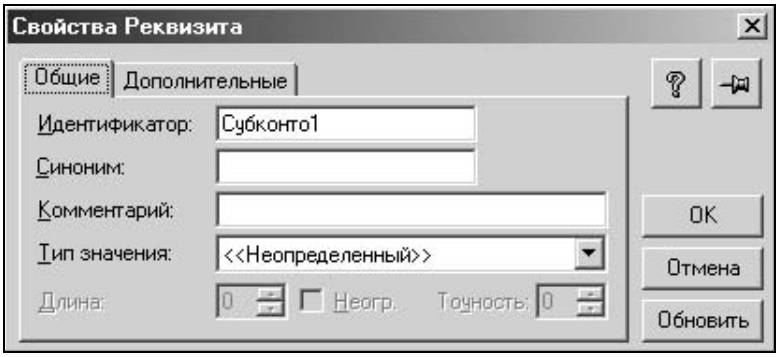


Рис. 3.12. Диалоговое окно свойств реквизита

Для объектов типа "Контекст", "Периодический", "Таблица" в методе определены следующие параметры:

- `ИмяТипа` — строковое выражение, определяющее название типа данных (или Вид Субконто), который назначается реквизиту (например, "Строка", "Число", "Справочник.Товары", "Документ.РасходнаяНакладная" и т. п.);
- `Длина` — необязательный параметр типа "Число", который определяет длину поля представления данных (имеет смысл только при задании числового или строкового типа);

- `Точность` — необязательный параметр, определяющий число знаков после десятичной точки (имеет смысл только при задании числового типа).

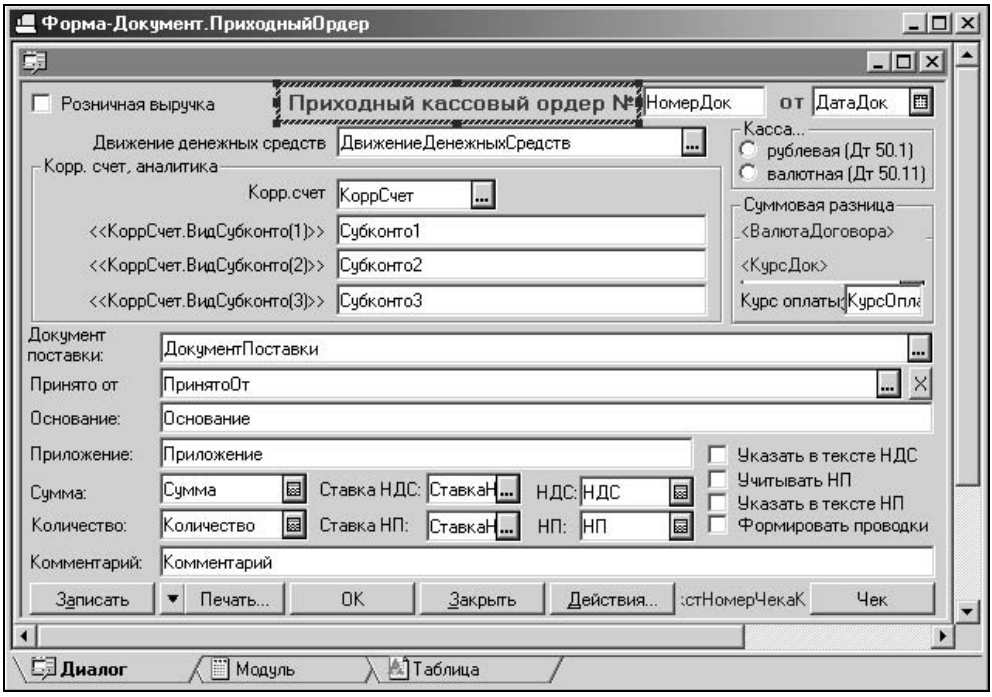


Рис. 3.13. Диалоговое окно формы документа **ПриходныйОрдер**

Наиболее характерным применением данного метода является назначение типов для реквизитов после выбора счета в соответствии с видами субконто выбранного счета. В документе "ПриходныйОрдер" реквизиты — **Субконто1**, **Субконто2** и **Субконто3** неопределенного типа (рис. 3.13). В модуле формы для назначения их типов определена процедура `ПриВыбореСчета()`:

```
Процедура ПриВыбореСчета()
   Для А = 1 По 3 Цикл
// Назначим тип реквизиту СубконтоА
            НазначитьТип("Субконто"+А,КоррСчет.ВидСубконто(А));
// Процедура ПриВыбореСубконто() рассматривается в методе
// ПолучитьАтрибут()
      ПриВыбореСубконто("Субконто"+А);
   КонецЦикла;
   ФлагВидимостиКоличества = 0;
   Если КоррСчет.Выбран() = 1 Тогда
```

```
    ФлагВидимостиКоличества = КоррСчет.Количественный;
  КонецЕсли;
  Форма.ПодписьКоличества.Видимость(ФлагВидимостиКоличества);
  Форма.Количество.Видимость(ФлагВидимостиКоличества);
  Если (Константа.ИспользоватьСписокКорректныхПроводок = Да) Тогда
    Если Валютный = 2 Тогда
      СчетКассы = СчетПоКоду("50.11");
    Иначе
      СчетКассы = СчетПоКоду("50.1");
    КонецЕсли;
    глПроверкаКорректныхПроводок(СчетКассы,КоррСчет);
  КонецЕсли;
КонецПроцедуры  // ПриВыбореСчета
```

2. `ПолучитьАтрибут()`.

Для получения значения атрибута по имени идентификатора, которое в явном виде не определено, предназначен метод `ПолучитьАтрибут(Параметр)`. Для разных объектов метаданных единственный `Параметр` определен поразному. Рассмотрим варианты значений, которые он может принимать.

- `ИмяРеквизита` — строка с наименованием реквизита, как он назван в конфигураторе для объектов метаданных типа "Документ", "Операция", "Регистр", "Справочник", "Счет", а также элемента формы для объекта метаданных типа "Контекст".
- `ИмяВидаРасч` — строковое выражение, содержащее имя конкретного вида расчета, как оно задано в конфигураторе для объектов метаданных типа "ВидРасчета".
- `ИмяАтрибута` — строковое выражение, содержащее имя атрибута, как оно задано в конфигураторе для объектов метаданных типа "Календарь", "Внешняя компонента V7Plus", "Запрос".
- `ИмяКалендаря` — строковое выражение, содержащее имя календаря конкретного вида, как оно задано в Конфигураторе для объектов метаданных типа "Календари".
- `ИмяКонстанты` — строковое выражение, содержащее идентификатор константы, как он задан в конфигураторе для объекта метаданных типа "Константа".
- `ИмяВидаПеречисл` — строковое выражение, содержащее имя перечисления, как оно задано в Конфигураторе для объектов типа "Перечисление".
- `ИмяПоследовательности` — строковое выражение, содержащее имя последовательности, как оно задано в конфигураторе для объектов типа "Последовательность".

- ИмяРегистра — строковое выражение, содержащее имя регистра, как оно задано в конфигураторе для объектов типа "Регистр".

3. УстановитьАтрибут().

Для задания значения атрибута по имени идентификатора, которое в явном виде не определено, предназначен метод УстановитьАтрибут(Параметр,Значение). Для разных объектов метаданных Параметр определен следующие образом:

- ИмяРеквизита — строка с наименованием реквизита, как он назван в Конфигураторе для объектов метаданных типа "Документ", "Операция", "Регистр", "Справочник", "Счет", а также элемента формы для объекта метаданных типа "Контекст";
- ИмяВидаРасч — строковое выражение, содержащее имя конкретного вида расчета, как оно задано в Конфигураторе для объектов метаданных типа "ВидРасчета";
- ИмяАтрибута — строковое выражение, содержащее имя атрибута, как оно задано в конфигураторе для объектов метаданных типа "Календарь", "Внешняя компонента V7Plus", "Запрос";
- ИмяКонстанты — строковое выражение, содержащее идентификатор константы, как он задан в конфигураторе для объекта метаданных типа "Константа";
- ИмяВидаПеречисл — строковое выражение, содержащее имя перечисления, как оно задано в Конфигураторе для объектов типа "Перечисление";
- ИмяПоследовательности — строковое выражение, содержащее имя последовательности, как оно задано в Конфигураторе для объектов типа "Последовательность";
- ИмяРегистра — строковое выражение, содержащее имя регистра, как оно задано в Конфигураторе для объектов типа "Регистр".

Параметр Значение — выражение, содержащее устанавливаемое значение атрибута.

В документе "ПриходныйОрдер" Реквизитам **Субконто1**, **Субконто2** и **Субконто3** назначена процедура ПриВыбореСубконто() для контроля ввода договора:

```
Процедура ПриВыбореСубконто(ТекЭлемент = "")
// Определим идентификатор реквизита, при выборе которого произошел
// вызов
   ИдентификаторРеквизита = ?(ТекЭлемент = "",
                              Форма.АктивныйЭлемент(), ТекЭлемент);
// Невозможно явно получить значение реквизита, поэтому
// воспользуемся методом
   ПолучитьАтрибут()
   ЗначениеРеквизита = ПолучитьАтрибут(ИдентификаторРеквизита);
```

```
// Получим уровень субконто счета
   НомерСубконто = Число(Прав(ИдентификаторРеквизита, 1));
   Если ТипЗначенияСтр(ЗначениеРеквизита) = "Справочник" Тогда
      Если ЗначениеРеквизита.Вид() = "Контрагенты" Тогда
        Если НомерСубконто < 3 Тогда
ИдентификаторСледующегоРеквизита = Лев(ИдентификаторРеквизита,
СтрДлина(ИдентификаторРеквизита)-1) + (НомерСубконто + 1);
ЗначениеСледующегоРеквизита =
ПолучитьАтрибут (ИдентификаторСледующегоРеквизита);
          Если ТипЗначенияСтр(ЗначениеСледующегоРеквизита) =
               "Справочник" Тогда
          Если ЗначениеСледующегоРеквизита.Вид() = "Договоры" Тогда
// Если договор не задан подставим значение реквизита
// ОсновнойДоговор с помощью метода УстановитьАтрибут()
              Если ЗначениеРеквизита <>
                 ЗначениеСледующегоРеквизита.Владелец Тогда
            УстановитьАтрибут     (ИдентификаторСледующегоРеквизита,
            ЗначениеРеквизита.ОсновнойДоговор);
              КонецЕсли;
            КонецЕсли;
          КонецЕсли;
        КонецЕсли;
        Если (НомерСубконто = 1) и (ПустоеЗначение(ПринятоОт) = 1)
      Тогда
          ПринятоОт = глПолноеНаименование(ЗначениеРеквизита);
        КонецЕсли;
      ИначеЕсли ЗначениеРеквизита.Вид() = "Договоры" Тогда
        Если (ПустоеЗначение(ЗначениеРеквизита) = 0)
        и (НомерСубконто > 1) Тогда
ИдентификаторПредыдущегоРеквизита = Лев(ИдентификаторРеквизита,
СтрДлина(ИдентификаторРеквизита)-1) + (НомерСубконто - 1);
ЗначениеПредыдущегоРеквизита=ПолучитьАтрибут
              (ИдентификаторПредыдущегоРеквизита);
        Если ТипЗначенияСтр(ЗначениеПредыдущегоРеквизита) =
        "Справочник" Тогда
          Если ЗначениеПредыдущегоРеквизита.Вид() = "Контрагенты" Тогда
            Если ЗначениеРеквизита.Владелец <>
                 ЗначениеПредыдущегоРеквизита Тогда
              УстановитьАтрибут(ИдентификаторПредыдущегоРеквизита,
                        ЗначениеРеквизита.Владелец);
```

```
            КонецЕсли;
          КонецЕсли;
        КонецЕсли;
      КонецЕсли;
      УправлениеВидимостьюСуммовыхРазниц();
      Если (ЗначениеРеквизита.Наименование =
           глИмяДоговораДляПлатежейБезДоговора) и
           (КоррСчет <> СчетПоКоду("62.2")) и
           (КоррСчет <> СчетПоКоду("62.22")) Тогда
        Предупреждение("Служебный договор ""Без договора..."""
                       можно использовать только при явно указанном
                       счете учета выданных авансов.
                       |Укажите другой договор.");
        УстановитьАтрибут(ИдентификаторРеквизита, "");
      КонецЕсли;
    ИначеЕсли ЗначениеРеквизита.Вид() = "Сотрудники" Тогда
Если (НомерСубконто = 1) и (ПустоеЗначение(ПринятоОт) = 1) Тогда
        ПринятоОт = ЗначениеРеквизита.Наименование;
      КонецЕсли;
    КонецЕсли;
  КонецЕсли;
КонецПроцедуры // ПриВыбореСубконто()
```

4. `Вид()`.

При обработке выборок, а также при анализе принятого параметра часто возникает необходимость определения вида текущего элемента справочника или документа. Для определения вида предназначен метод `Вид()`.

Метод без параметров и применяется для следующих объектов метаданных:

- Документ — возвращает строку с названием вида документа;
- ЖурналРасчетов — возвращает название журнала расчетов, как оно задано при конфигурировании журнала;
- Календарь — возвращает идентификатор календаря, заданный при конфигурировании;
- Перечисление — возвращает строку с названием вида перечисления;
- Регистр — возвращает строку с названием вида регистра;
- Справочник — возвращает текущее название вида справочника;
- Счет — возвращает символьную строку, определяющую идентификатор плана счетов, к которому относится данный счет.

Внимание

Для объектов метаданных типа "Справочник" определен параметр `Название` — строка с названием вида справочника. Если параметр задан, метод не только возвращает текущее название вида справочника, но и устанавливает новое заданное параметром название.

В приведенной выше процедуре данный метод использовался для определения вида справочников "Контрагенты" и "Договоры". Метод также используется в предопределенной процедуре `ВводНаОсновании()` при определении вида документа. В нашем документе "ПриходныйОрдер" процедура определена следующим образом:

```
Процедура ВводНаОсновании(ДокОсн)
   Новый = 1;
   ВерсияОбъекта = Константа.НомерРелиза;
   Если ДокОсн.Вид() = "ПродажаВРозницу" Тогда
      УказатьНДС = 0;
      УчитыватьНП = 0;
      УказатьНП = 0;
      ФормироватьПроводки = 0;
      Валютный = 1;
      Сумма = ДокОсн.Итог("Всего");
      СобственныхТоваров = 0;
      КомиссионныхТоваров = 0;
      ДокОсн.ВыбратьСтроки();
      Пока ДокОсн.ПолучитьСтроку() = 1 Цикл
Если ДокОсн.Товар.ТипТовара = Перечисление.ТипыТоваров.НаКомиссии
         Тогда
            КомиссионныхТоваров = КомиссионныхТоваров + 1;
         Иначе
            СобственныхТоваров = СобственныхТоваров + 1;
         КонецЕсли
      КонецЦикла;
      Если СобственныхТоваров = ДокОсн.КоличествоСтрок() Тогда
         Если ДокОсн.ПродажаОблагаетсяЕНВД = 0 Тогда
            КоррСчет = СчетПоКоду("90.1.1");
         Иначе
            КоррСчет = СчетПоКоду("90.1.2");
         КонецЕсли;
      ИначеЕсли КомиссионныхТоваров = ДокОсн.КоличествоСтрок() Тогда
```

```
        КоррСчет = СчетПоКоду("76.5");
    КонецЕсли;
    ПриВыбореСчета();
КонецЕсли;
Комментарий = "Введен на основании:
                              "+глПредставлениеДокумента(ДокОсн);
КонецПроцедуры  // ВводНаОсновании()
```

5. `ПредставлениеВида()`.

Для отображения наименования и ряда значений атрибутов объектов метаданных в виде, понятном пользователю, предназначен метод `ПредставлениеВида()`. Метод без параметров и применяется для следующих объектов метаданных:

- Документ — возвращает строку с названием вида документа;
- ЖурналРасчетов — возвращает название журнала расчетов, как оно задано при конфигурировании журнала;
- Календарь — возвращает идентификатор календаря, заданный при конфигурировании;
- Перечисление — возвращает строку с названием вида перечисления;
- Регистр — возвращает строку с названием вида регистра;
- Справочник — возвращает текущее название вида справочника;
- Счет — возвращает: символьную строку, определяющую идентификатор плана счетов, к которому относится данный счет.

Для примера рассмотрим функцию глобального модуля `глКонтрольДаты Документа()`, в которой, в сообщении пользователю о смене нумерации, сформировано название документа в виде, понятном пользователю:

```
Функция глКонтрольДатыДокумента(Конт, НачальнаяДатаДокумента) Экспорт
Перем ФлагПрисвоенияНомера; // для возвращаемого значения
ФлагПрисвоенияНомера=0;
ПериодСменыНомера = Метаданные.Документ( Конт.Вид()).ПериодичностьНомера;
Если ПериодСменыНомера = "Год" Тогда
РазностьДат = НачГода(НачальнаяДатаДокумента)-НачГода(Конт.ДатаДок);
ИначеЕсли ПериодСменыНомера = "Квартал" Тогда
    РазностьДат = НачКвартала(НачальнаяДатаДокумента) -
                  НачКвартала(Конт.ДатаДок);
ИначеЕсли ПериодСменыНомера = "Месяц" Тогда
   РазностьДат=НачМесяца(НачальнаяДатаДокумента)-
    НачМесяца(Конт.ДатаДок);
ИначеЕсли ПериодСменыНомера = "День" Тогда
   РазностьДат = НачальнаяДатаДокумента-Конт.ДатаДок;
```

```
Иначе
   РазностьДат = 0;
КонецЕсли;
Если РазностьДат <> 0 Тогда
   Дубликат = СоздатьОбъект("Документ."+Конт.Вид());
   Дубликат.Новый();
   Дубликат.НомерДок = Конт.НомерДок;
   Дубликат.ДатаДок = Конт.ДатаДок;
   Дубликат.УстановитьНовыйНомер("");
Если ( Дубликат.НомерДок <> Конт.НомерДок )
Тогда
    НовыйНомерДок = Дубликат.НомерДок;
Если Дубликат.НайтиПоНомеру ( Дубликат.НомерДок,Дубликат.ДатаДок) =0
     Тогда
Если Вопрос( "Для документов вида """ + Конт.ПредставлениеВида() +
      """ каждый " + Нрег(ПериодСменыНомера) + " нумерация
      начинается заново."+РазделительСтрок +"Присвоить новый
      номер?","Да+Нет") = "Да"
       Тогда
          Конт.НомерДок = НовыйНомерДок;
          ФлагПрисвоенияНомера = 1;
       КонецЕсли;
    КонецЕсли;
   КонецЕсли;
   НачальнаяДатаДокумента = Конт.ДатаДок;
   Конт.Активизировать("НомерДок");
КонецЕсли;
Возврат ФлагПрисвоенияНомера;
КонецФункции // глКонтрольДатыДокумента
```

6. `Выбран()`.

Данный метод предназначен для определения факта выбора объекта метаданных. Метод возвращает флаг выбора элемента справочника, принимающего значения:

- 1 — если элемент справочника выбран;
- 0 — если элемент справочника не выбран.

Метод определен для объектов метаданных типа "ВидРасчета", "ВидСубконто", "Документ", "Календарь", "Перечисление", "ПланСчетов", "Справочник",

"Счет", а также проверяет, спозиционирован ли объект типа "Метаданные" на конкретном объекте метаданных или нет. Возвращает значение — 1, если объект соответствует объекту метаданных (спозиционирован). 0 — в противном случае.

Введем в нашем документе "ПоказанияВодомеров" контроль задания реквизита **Услуга** в предопределенной процедуре `ПриЗаписи()`:

```
Процедура ПриЗаписи()
  Если Услуга.Выбран()=0 тогда
Предупреждение("Не выбран вид расчета - услуга! Документ не записан",30);
    СтатусВозврата(0);
    Возврат;
  КонецЕсли;
  Кв = СоздатьОбъект("Справочник.Квартиры");
  ВыбратьСтроки();
  Пока ПолучитьСтроку()=1 цикл
    Если Кв.НайтиПоКоду(Квартира.Код,0) = 1 тогда
Кв.ПоказаниеХолВоды.Установить(КонМесяца(ДатаДок),
                                          ПоказаниеСчетчика);
     Кв.Записать();
    КонецЕсли;
  КонецЦикла;
КонецПроцедуры
```

7. `Блокировка()`.

Метод `Блокировка(ВклВыкл)` предназначен для определения блокировки объекта, а также для установки блокировки обрабатываемого объекта. Если при вызове метода параметр `ВклВыкл` не задан, то возвращается значение режима блокировки до выполнения метода:

- 1 — заблокирован;
- 0 — свободен.

Если при вызове метода параметр `ВклВыкл` задан, то возвращается результат выполнения метода блокировки:

- 1 — успешно (True);
- 0 — не получилось (False).

Метод определен для объектов метаданных типа "Документ", "Справочник", "Счет".

Методы с использованием функции *СоздатьОбъект()*

В предыдущих разделах данной главы мы рассматривали методы объектов, которые были уже спозиционированы, указатель находился на конкретном объекте. В случае неопределенности методом `Выбран()` можно было определить, спозиционирован объект или нет. Если же необходимо найти какой-либо объект или группу объектов или создать новый объект, то в самом начале надо создать указатель на объект. Специальная функция `СоздатьОбъект()` создает именно такой указатель. Обычная схема использования методов этой группы представлена на рис. 3.14.

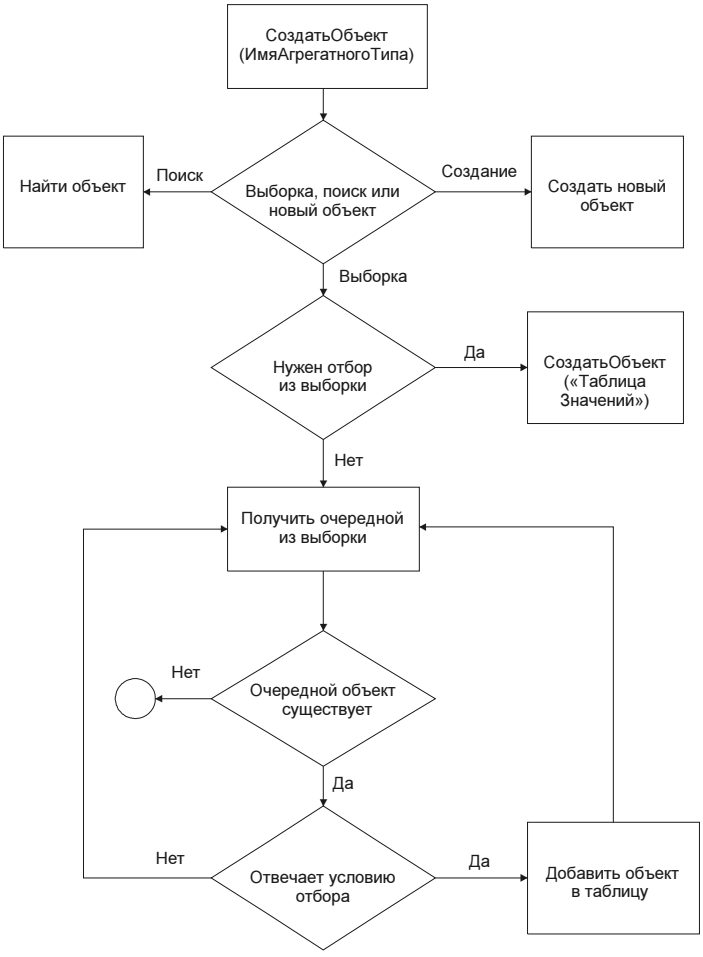


Рис. 3.14. Схема использования методов

Функция *СоздатьОбъект()*

1. `СоздатьОбъект()`.

Функция `СоздатьОбъект(ИмяАгрегатногоТипа)` создает указатель на объект агрегатного типа, т.е. на объект метаданных. Параметр `ИмяАгрегатногоТипа` — строковое выражение, содержащее имя агрегатного типа данных, объявленного в Конфигураторе. Строка может быть такой, какой она представлена в списке, предложенной на выбор в поле **Тип значения** диалогового окна **Свойства реквизита** (см. рис. 3.15). При задании значения параметра `ИмяАгрегатногоТипа` "Справочник" или " Документ" создается указатель для всех видов справочников или документов. Выборка для таких объектов будет содержать все виды справочников или документов.

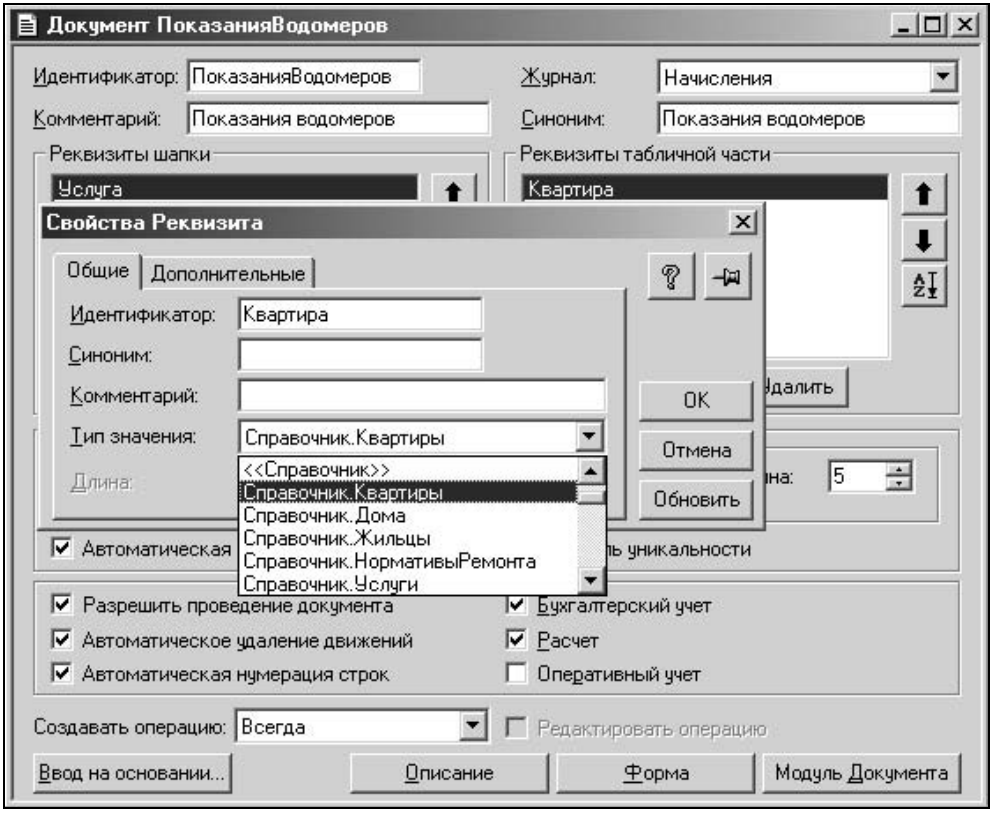


Рис. 3.15. Диалоговое окно **Свойства Реквизита**

Вернемся еще раз к функции `ОпределениеВладельца()` в модуле формы списка справочника " Квартиры":

```
Функция ОпределениеВладельца ()
    // Создадим указатель на объект типа «Справочник.Жильцы»
```

```
    Жильцы = СоздатьОбъект("Справочник.Жильцы");
    // Зададим выборку элементов справочника по владельцу
    Жильцы.ИспользоватьВладельца(ТекущийЭлемент());
    // Откроем выборку
    Жильцы.ВыбратьЭлементы();
    // Просмотрим элементы выборки
    Пока Жильцы.ПолучитьЭлемент()=1 цикл
      // Если указатель позиционирован на искомого владельца квартиры
      Если Жильцы.СтатусПрописки = Перечисление.Статусы.Владелец тогда
      // Возврат значения атрибута ФИО
              Возврат Жильцы.ФИО
       КонецЕсли;
    КонецЦикла;
КонецФункции
```

В данной функции владелец определялся перебором всех элементов конкретного подчиненного справочника "Жильцы".

В справочнике "Договоры" в модуле формы списка `ДляВыбора` для кнопки **Создать по счету** определена процедура, где также создается объект. Текст процедуры приводится ниже.

```
Процедура СоздатьПоСчету()
  Перем Договор;
//  Создадим указатель на объект типа "Справочник.Договоры"
  Спр = СоздатьОбъект("Справочник.Договоры");
// Зададим выборку элементов справочника по владельцу
  Спр.ИспользоватьВладельца(Владелец);
//  Создадим указатель на объект типа «СписокЗначений»
  Счета = СоздатьОбъект("СписокЗначений");
//  Создадим указатель на объект типа «Документ»
  Док = СоздатьОбъект("Документ");
// Зададим выборку документов владельца всех видов
  Док.ВыбратьПоЗначению(,,"Контрагент",Владелец);
// Просмотрим все выбранные документы и отберем документы
//вида "Счет"
  Пока Док.ПолучитьДокумент() = 1 Цикл
    Если Док.Вид() = "Счет" Тогда
       НаименованиеДоговора = "Счет №"+Док.НомерДок+" от " + Формат
                                    ( Док.ДатаДок,"Д ДД.ММ.ГГ");
// Проверим существование данного договора в отобранных элементах
// справочника
    Если Спр.НайтиПоНаименованию(НаименованиеДоговора) = 0 Тогда
```

```
// Добавим в список для выбора наименование договора по счету
        Счета.ДобавитьЗначение(НаименованиеДоговора);
      КонецЕсли;
    КонецЕсли;
  КонецЦикла;
// Предложим форму для выбора наименования для договора
// из сформированного списка
  Если Счета.ВыбратьЗначение(Договор, "Выберите счет") = 1 Тогда
// Создадим новый договор с выбранным наименованием
    Спр.Новый();
    Спр.Наименование = Договор;
    Спр.ДатаВозникновенияОбязательства =
                                    Дата(Сред(Договор, 13, 21));
    Спр.ОплатаДоговора = 1;
// Запишем созданный договор
    Спр.Записать();
// Выберем созданный договор в Фоме вызова.
    Форма.ВыполнитьВыбор(Спр.ТекущийЭлемент());
  КонецЕсли;
КонецПроцедуры // СоздатьПоСчету()
```

2. `Новый()`.

Данный метод добавляет новый объект метаданных. Метод определен для объектов метаданных типа "Справочник", " Документ", " Счет".

В предыдущем разделе, в процедуре `СоздатьПоСчету()`, был использован данный метод создания нового элемента справочника "Договоры".

Внимание

Методом `Новый()` на самом деле создается новый объект только в памяти. Для записи объекта в базу данных, после его создания, необходимо применить метод `Записать()`.

3. `Новая()`.

Данный метод добавляет новый объект метаданных. Метод определен для объектов метаданных типа "ЖурналРасчетов" — для создания новой строки в журнал расчетов, "КорректныеПроводки" — для создания новой корректной проводки, "Операции" — для создания новой операции.

В разделе "Атрибуты операций и проводок" была приведена процедура `ОбработкаПроведения()` модуля документа "Показания водомеров", в которой использовался данный метод:

```
ЖурналРасчетовНачислений.Новая();
```

Внимание

Методом `Новая()` для операций создается только операция. Для создания проводок текущей операции необходимо использовать метод `НоваяПроводка()`. В модуле документа, в предопределенной процедуре `ОбработкаПроведения()`, операция для текущего документа уже создана, если в поле **Создавать операцию** задано соответствующее значение (см. рис. 3.16). Использовать метод `Записать()`, для записи операции, следует после задания всех проводок.

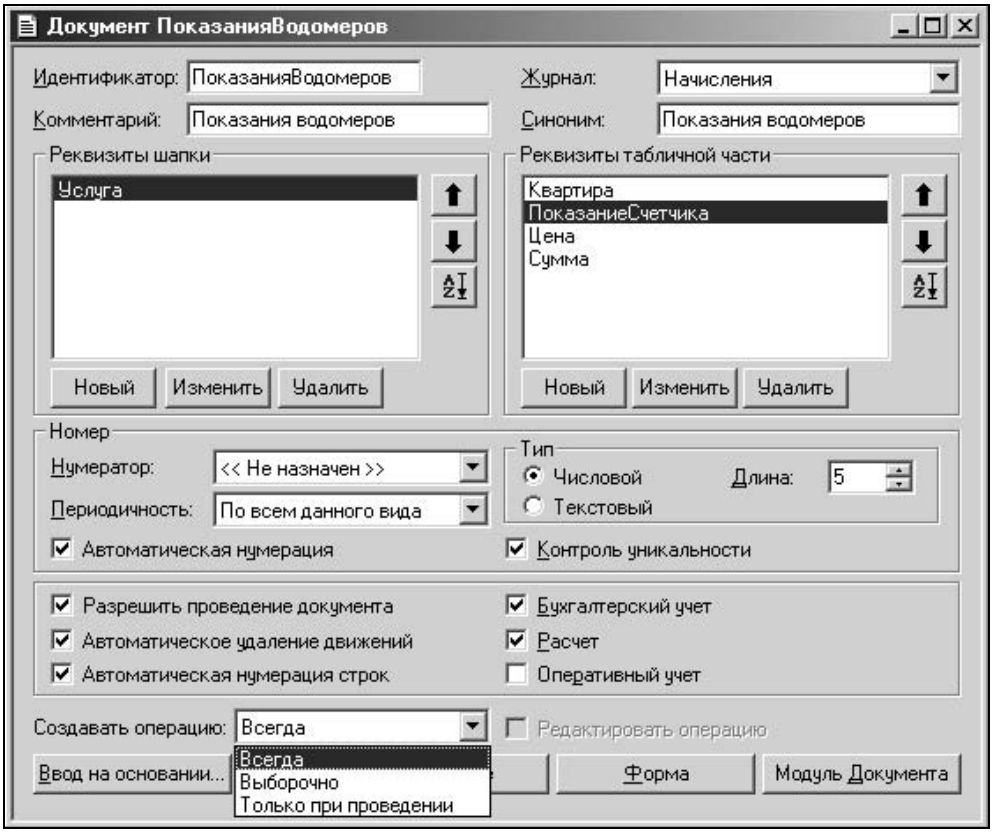


Рис. 3.16. Диалоговое окно документа

4. `НоваяПроводка()`.

Данный метод создает новую проводку для текущей операции. Новая проводка становится текущей. В разделе "Атрибуты операций и проводок" была приведена процедура `ОбработкаПроведения()` модуля документа "Показания водомеров", в которой использовался данный метод:

```
Операция.НоваяПроводка();
```

5. Записать().

Данный метод предназначен для записи изменений в базу данных.

Возвращает:

- 1 — если вызов метода закончился успешно;
- 0 — в противном случае.

Метод определен для объектов метаданных типа "Документ", "Справочник", "Счет", "Журнал расчетов", "Корректные проводки", "Операции", "Таблица", "Текст". Метод Записать(), примененный в модуле формы документа непосредственно к документу локального контекста, отрабатывает те же действия, как интерактивное нажатие пользователем кнопки с формулой "#Записать". Метод, примененный в модуле формы операции непосредственно к операции локального контекста, отрабатывает те же действия, что и интерактивное нажатие пользователем кнопки с формулой "#Записать". При записи сложной проводки, если у главной корреспонденции сложной проводки не указана сумма (равна 0), то она автоматически вычисляется на основании подчиненных корреспонденций. Метод, примененный в модуле формы элемента справочника непосредственно к элементу справочника локального контекста, отрабатывает те же действия, что и интерактивное нажатие пользователем кнопки с формулой "#Записать". Метод, примененный в модуле формы счета непосредственно к счету локального контекста, отрабатывает те же действия, что и интерактивное нажатие пользователем кнопки с формулой "#Записать".

Метод вносит изменения записи или новую запись в журнал расчетов. Данный метод применяется после метода Новая() и заполнения реквизитов журнала расчетов при помощи метода УстановитьРеквизит(). Метод Записать() проверяет корректность заполненных реквизитов журнала расчетов. При вводе новых записей журнала расчетов методами Новая() и Записать() обязательно должны быть заполнены некоторые реквизиты. При вводе новых записей в журнал расчетов методами Новая() и Записать(), записи вводятся так, как они есть. Система не выполняет правила перерасчетов, а также правила взаимного вытеснения видов расчета. В разделе "Атрибуты операций и проводок" была приведена процедура ОбработкаПроведения() модуля документа "Показания водомеров", в которой использовался данный метод для записи в журнал расчетов в фрагменте:

```
ЖурналРасчетовНачислений.Новая();
ЖурналРасчетовНачислений.УстановитьРеквизит
("Объект",Квартира);
ЖурналРасчетовНачислений.УстановитьРеквизит
("ВидРасч",Услуга.Расчет);
ЖурналРасчетовНачислений.УстановитьРеквизит
("Документ", ТекущийДокумент());
```

```
ЖурналРасчетовНачислений.УстановитьРеквизит
("РодительскийДокумент",ТекущийДокумент());
ЖурналРасчетовНачислений.УстановитьРеквизит
("ДатаНачала",ДатаН);
ЖурналРасчетовНачислений.УстановитьРеквизит
("ДатаОкончания",ДатаК);
ЖурналРасчетовНачислений.УстановитьРеквизит
("НомерСтрокиДокумента",НомерСтроки);
ЖурналРасчетовНачислений.Записать();
ЖурналРасчетовНачислений.Рассчитать();
```

6. `Выбрать()`.

При нажатии кнопки выбора элемента диалога, для которого назначен тип "Справочник", на экран выводится форма списка справочника для выбора элемента. Аналогично выбирается документ из журнала и счет из плана счетов. Метод `Выбрать()` предназначен для вызова формы выбора. Он возвращает:

- 1 — если элемент выбран;
- 0 — в противном случае.

Для объектов метаданных типа "Справочник" в методе определены следующие параметры:

- `Подсказка` — текст заголовка окна диалога ввода;
- `ФормаСписка` — строка, содержащая идентификатор формы списка справочника, используемой для выбора.

Для объектов метаданных типа "Документ" в методе определен также ряд параметров. Рассмотрим их по отдельности.

- `Подсказка` — текст заголовка окна диалога ввода.
- `ФормаЖурнала` — строка, содержащая идентификатор формы журнала документа, используемой для выбора. Здесь можно указывать имя объекта в следующем виде: "Журнал.XXXXX,YYYYY", где XXXXX — идентификатор журнала (как он задан в конфигураторе), YYYYY — имя формы журнала. Кроме этого допускается указание в формах — "Журнал.Подчиненные"; "ЖурналОпераций"; "ЖурналОпераций.YYYYY", где YYYYY будет означать имя формы журнала операций (как оно задано в Конфигураторе). Для пустой строки используется форма журнала по умолчанию.
- `КомуПодч` — необязательный параметр. Используется при открытии выбора по журналу подчиненных документов, т. е. когда второй параметр `ФормаЖурнала` имеет значение `Журнал.Подчиненные`. В данном случае в этом параметре передается документ-владелец, по которому следует построить журнал подчиненных документов. Для пустой строки используется форма журнала по умолчанию.

Метод, как правило, используется в случаях, когда для выбора необходимо предложить не весь справочник или журнал документов, а выборку из них, а также для реквизитов неопределенного типа. Например, в документе "Выписка" для задания первичного документа формируется выборка документов, которые будут предложены для выбора пользователю:

```
Процедура ПоступлениеПоДокументам()
  Меню = СоздатьОбъект("СписокЗначений");
  Меню.ДобавитьЗначение("Счет", "Счет");
  Меню.ДобавитьЗначение("СчетФактура", "Счет-фактура");
  Меню.ДобавитьЗначение("РасходнаяНакладная", "Отгрузка товаров,
                                                продукции");
  Меню.ДобавитьЗначение("РеализацияОтгруженнойПродукции",
                        "Реализация отгруженной продукции");
  Меню.ДобавитьЗначение("ОказаниеУслуг", "Оказание услуг");
  Меню.ДобавитьЗначение("ВыполнениеЭтапаРабот" , "Выполнение
                                                  этапа работ");
  Меню.ДобавитьЗначение("ОтпускМатериаловНаСторону", "Отгрузка
                                          материалов на сторону");
  Меню.ДобавитьЗначение("ПередачаОС", "Передача ОС");
  Меню.ДобавитьЗначение("ПередачаНМА", "Передача НМА");
  ВидДок = "";
  Если Меню.ВыбратьЗначение(ВидДок,,,,1) = 1 Тогда
     Док = СоздатьОбъект("Документ."+ВидДок);
// Предложим диалоговое окно для выбора документа
     Если Док.Выбрать("Выберите документ, в оплату которого
                       поступили денежные средства") = 1 Тогда
       СформироватьСтрокиПоДокументу(Док);
     КонецЕсли;
  КонецЕсли;
КонецПроцедуры  // ПоступлениеПоДокументам
```

7. `Удалить()`.

Метод `Удалить (Режим)` предназначен для удаления объекта метаданных. Параметр `Режим` задает режим удаления. Он принимает значения:

- 1 — непосредственное удаление;
- 0 — пометка на удаление.

Метод определен для объектов метаданных типа "Документ", "Справочник", "Счет", "Корректные проводки", "Операции", "Таблица", "Текст", "Календарь".

8. `ПометкаУдаления()`.

Метод проверяет наличие пометки на удаление объекта метаданных. Он возвращает:

- 1 — операция помечена на удаление;
- 0 — операция не помечена на удаление.

Метод определен для объектов метаданных типа "Справочник", "Документ", "Операция", "Счет".

Данный метод используется для того, чтобы исключить из обработки объекты, помеченные на удаление. Например:

```
Док = СоздатьОбъект(«Документ»);
Док.ВыбратьДокументы(,);
Пока Док.ПолучитьДокумент()=1 Цикл
// Проверим наличие пометки удаления
    Если Док.ПометкаУдаления()=1 Тогда
      Продолжить;
    КонецЕсли;
    КонецЦикла;
```

9. `СнятьПометкуУдаления()`.

Метод снимает пометку удаления с объекта метаданных. Метод определен для объектов метаданных типа "Справочник", "Документ", "Операция", "Счет".

Рассмотрим пример, где снимаются все пометки удаления с документов:

```
Док = СоздатьОбъект(«Документ»);
Док.ВыбратьДокументы(,);
Пока Док.ПолучитьДокумент()=1 Цикл
// Проверим наличие пометки удаления
    Если Док.ПометкаУдаления()=1 Тогда
// Снимем пометку удаления
      Док.СнятьПометкуУдаления();
    КонецЕсли;
КонецЦикла;
```

10. `ОбратныйПорядок()`.

Метод `ОбратныйПорядок(Режим)` устанавливает порядок выборки документов во времени. Возвращает значение порядка выборки до вызова:

- 1 — выборка объектов в обратном порядке;
- 0 — выборка объектов в прямом порядке.

Параметр — Режим является необязательным параметром. Он принимает значения:

- 1 — выбирать объекты в обратном порядке;
- 0 — выбирать объекты в прямом порядке.

Если параметр опущен, то метод просто возвращает текущее значение порядка выборки документов.

Данный метод определен для объектов типа "Документ", " Справочник", "Периодический", " Регистр".

Методы документов

1. НайтиДокумент().

Метод НайтиДокумент(Документ) предназначен для нахождения документа по значению (или позиционировать документ). Возвращает:

- 1 — если действие выполнено (документ найден);
- 0 — если действие не выполнено.

Параметр Документ — выражение со значением типа "Документ".

2. НайтиДокументПоНомеру().

Метод НайтиДокументПоНомеру(Номер,Дата,ИдентификаторВида) предназначен для поиска документа по номеру. Он может возвращать значения:

- 1 — если действие выполнено (документ найден);
- 0 — если действие не выполнено.

Параметры:

- Номер — строка с номером искомого документа;
- Дата — дата из диапазона, в котором нужно искать документ.
- ИдентификаторВида — необязательный параметр (строковое выражение, содержащее ИдентификаторВида документа или идентификатор Нумератора).

Параметр **ИдентификаторВида** задается для объекта "Документ" общего вида.

3. ВыбратьДокументы().

Метод ВыбратьДокументы(Дата1,Дата2) открывает выборку документов в интервале дат.

Возвращает значения:

- 1 — если действие выполнено и в выборке есть хотя бы один документ;
- 0 — если действие не выполнено или в выборке нет ни одного документа.

Параметры:

- `Дата1` — дата, документ или позиция начала выборки документов (если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа);
- `Дата2` — дата, документ или позиция конца выборки документов (если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом).

Снимем пометки на удаление с документов за период с 1 января 2004 г. по 31 марта 2004 г.:

```
Док = СоздатьОбъект(«Документ»);
// Зададим выборку документов
Док.ВыбратьДокументы ("01.01.2004","31.03.2004");
Пока Док.ПолучитьДокумент()=1 Цикл
    Если Док.ПометкаУдаления()=1 Тогда
      Док.СнятьПометкуУдаления();
    КонецЕсли;
КонецЦикла;
```

4. `ВыбратьПодчиненныеДокументы()`.

Подчиненными документами являются документы, которые были введены на основании другого документа, который в свою очередь называется документом-основанием.

Метод `ВыбратьПодчиненныеДокументы(Дата1,Дата2,Документ)` открывает выборку документов, подчиненных заданному документу-основанию, в интервале дат.

Возвращает значения:

- 1 — если действие выполнено и в выборке есть хотя бы один документ;
- 0 — если действие не выполнено или в выборке нет ни одного документа.

Параметры:

- `Дата1` — дата, документ или позиция начала выборки документов (если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа);
- `Дата2` — дата, документ или позиция конца выборки документов (если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом);
- `Документ` — документ-основание.

Рассмотрим использование метода на примере. Для этого создадим в журнале "Ремонт" графу `Оплата` с типом "Текст", без привязки к реквизиту

документов. Для отображения суммы оплаты документа определим для этой графы функцию Оплата():

```
Функция Оплата()
  Сумма=0;
  Если (ТекущийДокумент.Вид() = "Ремонт") Тогда
     ТекущийДокумент.Операция.ВыбратьПроводки();
     Пока ТекущийДокумент.Операция.ПолучитьПроводку()=1 Цикл
       Если ТекущийДокумент.Операция.Дебет.Счет = СчетПоКоду("62.2")
Тогда
          Сумма = Сумма+ТекущийДокумент.Операция.Сумма;
       КонецЕсли;
      КонецЦикла;
     Если Сумма=0 Тогда
       Док=СоздатьОбъект("Документ");
       Док.ВыбратьПодчиненныеДокументы(,,ТекущийДокумент);
       Пока Док.ПолучитьДокумент()=1 Цикл
          Если Док.Вид()="Выписка" Тогда
            Док.ВыбратьСтроки();
            Пока Док.ПолучитьСтроку()=1 Цикл
              Если Док.ДокументПоставки=ТекущийДокумент Тогда
                 Сумма = Док.Приход;
              КонецЕсли;
            КонецЦикла;
          ИначеЕсли Док.Вид()="ПриходныйОрдер" Тогда
            Сумма = Док.Сумма;
          КонецЕсли;
       КонецЦикла;
```

5. ВыбратьПоЗначению().

Метод ВыбратьПоЗначению(Дата1,Дата2,ИмяОтбора,Значение) открывает выборку документов в интервале дат с заданным значением реквизита отбора.

Возвращает:

- 1 — если действие выполнено и в выборке есть хотя бы один документ;
- 0 — если действие не выполнено или в выборке нет ни одного документа.

Параметры:

- Дата1 — дата, документ или позиция начала выборки документов (если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа);

- `Дата2` — дата, документ или позиция конца выборки документов (если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом);
- `ИмяОтбора` — строка с названием общего реквизита документов либо названием графы отбора журналов;
- `Значение` — значение отбора, по которому строится выборка документов.

Изменим процедуру `ПоступлениеПоДокументам()`, приведенную для иллюстрации метода `Выбрать()`. Для этого используем метод `ВыбратьПоЗначению()` для сокращения выборки до перечня документов только заданного контрагента.

```
Процедура ПоступлениеПоДокументам()
  Меню = СоздатьОбъект("СписокЗначений");
  Меню.ДобавитьЗначение("Счет", "Счет");
  Меню.ДобавитьЗначение("СчетФактура", "Счет-фактура");
  Меню.ДобавитьЗначение("РасходнаяНакладная", "Отгрузка товаров,
                                              продукции");
  Меню.ДобавитьЗначение("РеализацияОтгруженнойПродукции",
                        "Реализация отгруженной продукции");
  Меню.ДобавитьЗначение("ОказаниеУслуг", "Оказание услуг");
  Меню.ДобавитьЗначение("ВыполнениеЭтапаРабот", "Выполнение этапа
                                                  работ");
  Меню.ДобавитьЗначение("ОтпускМатериаловНаСторону", "Отгрузка
                                    материалов на сторону");
  Меню.ДобавитьЗначение("ПередачаОС", "Передача ОС");
  Меню.ДобавитьЗначение("ПередачаНМА", "Передача НМА");
  ВидДок = "";
  Если Меню.ВыбратьЗначение(ВидДок,,,,1) = 1 Тогда
     Док = СоздатьОбъект("Документ."+ВидДок);
// Контрагент задан в поле Субконто1, ограничим выборку по Субконто1
     Док.ВыбратьПоЗначению(,ДатаДок,«Контрагент»,Субконто1);
// Предложим диалоговое окно для выбора документа из перечня
// документов заданного контрагента
     Если Док.Выбрать("Выберите документ, в оплату которого поступили
                       денежные средства") = 1 Тогда
       СформироватьСтрокиПоДокументу(Док);
     КонецЕсли;
  КонецЕсли;
КонецПроцедуры // ПоступлениеПоДокументам
```

6. `ВыбратьПоНомеру()`.

Метод `ВыбратьПоНомеру(Номер,Дата,ИдентВида)` открывает выборку документов в интервале дат по номеру.

Возвращает значения:

- 1 — если действие выполнено и в выборке есть хотя бы один документ;
- 0 — если действие не выполнено или в выборке нет ни одного документа.

Параметры:

- `Номер` — строка, содержащая номер искомых документов;
- `Дата` — любая дата из диапазона, в котором нужно искать документ с данным номером. Поиск зависит от выбранного в Конфигураторе способа уникальности номеров (по месяцу, году и др.);
- `ИдентВида` — необязательный параметр (строковое выражение, содержащее идентификатор вида документа или идентификатор "Нумератора").

7. `ВыбратьПоПоследовательности()`.

Метод `ВыбратьПоПоследовательности(Дата1,Дата2,КодПоследовательности)` открывает выборку документов в интервале дат по заданной последовательности. Возвращает значения:

- 1 — если действие выполнено и в выборке есть хотя бы один документ;
- 0 — если действие не выполнено или в выборке нет ни одного документа.

Параметры:

- `Дата1` — необязательный параметр, представляющий дату, документ или позицию начала выборки документов (если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа);
- `Дата2` — необязательный параметр, представляющий дату, документ или позицию конца выборки документов (если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом);
- `КодПоследовательности` — строка с названием используемой последовательности.

8. `УстановитьФильтр()`.

Метод `УстановитьФильтр(Проведенные,Непроведенные,НеИмеющиеПризнаковУчета, Оперативные,Расчетные,Бухгалтерские)` назначает фильтр выборки документов.

Параметры:

- `Проведенные` — число, принимающее значения:
  - 0 — не включать в выборку проведенные документы;
  - 1 — включать;

- ❒ `Непроведенные` — число, принимающее значения:
  - 0 — не включать в выборку непроведенные документы;
  - 1 — включать;
- ❒ `НеИмеющиеПризнаковУчета` — число, принимающее значения:
  - 0 — не включать в выборку документы, не имеющие признаков учета;
  - 1 — включать;
- ❒ `Оперативные` — число, принимающее значения:
  - 0 — не включать в выборку оперативные документы;
  - 1 — данный флаг не влияет на выборку;
  - 2 — если оперативный документ, то включается в выборку;
- ❒ `Расчетные` — число, принимающее значения:
  - 0 — не включать в выборку расчетные документы;
  - 1 — данный флаг не влияет на выборку;
  - 2 — если расчетный документ, то включается в выборку;
- ❒ `Бухгалтерские` — число, принимающее значения:
  - 0 — не включать в выборку бухгалтерские документы;
  - 1 — данный флаг не влияет на выборку;
  - 2 — если бухгалтерский документ, то включается в выборку.

9. `ПолучитьДокумент()`.

Метод `ПолучитьДокумент()` получает из выборки следующий документ. Возвращает значения:

- ❒ 1 — если документ выбран;
- ❒ 0 — в противном случае.

Метод не имеет параметров.

При иллюстрации метода `ВыбратьДокументы()` мы использовали данный метод для перебора всех документов, заданных в выборке:

```
Док = СоздатьОбъект("Документ");
// Зададим выборку документов
Док.ВыбратьДокументы ("01.01.2004","31.03.2004");
// Выбираем следующий документ, пока метод возвращает 1
Пока Док.ПолучитьДокумент()=1 Цикл
    Если Док.ПометкаУдаления()=1 Тогда
      Док.СнятьПометкуУдаления();
    КонецЕсли;
КонецЦикла;
```

10. УстановитьВремя().

Метод УстановитьВремя(Часы,Минуты,Секунды) устанавливает время документа.

Параметры:

- Часы — число часов;
- Минуты — число минут;
- Секунды — число секунд.

11. ПолучитьВремя().

С помощью метода ПолучитьВремя(Часы,Минуты,Секунды) можно прочитать время создания документа. Возвращает время документа в переданные для этого переменные Часы, Минуты, Секунды.

Возвращает значение времени записи документа в виде строки в форме — "ЧЧ.ММ.СС".

Параметры:

- Часы — переменная для приема часа записи документа;
- Минуты — переменная для приема минут записи документа;
- Секунды — переменная для приема секунд записи документа.

12. СделатьНеПроведенным().

Метод СделатьНеПроведенным() отменяет проведение документа.

Внимание

Метод нельзя использовать в теле предопределенной процедуры ОбработкаПроведения().

Методы справочников

1. НайтиЭлемент().

Метод НайтиЭлемент(Элемент) находит элемент справочника по значению. Метод выполняет поиск элемента справочника и позиционирует объект на этом элементе.

Возвращает значения:

- 1 — если действие выполнено;
- 0 — если действие не выполнено (элемент не найден).

Параметр — Элемент, представляет выражение со значением элемента справочника.

В модуле формы документа "Показания водомеров", в предопределенной процедуре ПриЗаписи(), в справочник "Квартиры" записывается в историю

реквизита показание водомера на текущий месяц. Поскольку метод `Установить()` определен для объектов, созданных методом `СоздатьОбъект()`, необходимо позиционировать указатель на элементе справочника. Это происходит следующим образом:

```
Процедура ПриЗаписи()
    Если Услуга.Выбран()=0 тогда
      Сообщить("Не выбран вид расчета – услуга! Документ не записан");
      Предупреждение("Не выбран вид расчета – услуга! Документ
не записан",30);
      СтатусВозврата(0);
      Возврат;
    КонецЕсли;
// Для записи показания водомера в историю создадим указатель Кв
    Кв = СоздатьОбъект("Справочник.Квартиры");
    ВыбратьСтроки();
    Пока ПолучитьСтроку()=1 цикл
//  Позиционируем указатель на элемент справочника
      Если Кв.НайтиЭлемент(Квартира) = 1 тогда
         Кв.ПоказаниеХолВоды.Установить(КонМесяца(ДатаДок),
                                        ПоказаниеСчетчика);
         Кв.Записать();
      КонецЕсли;
    КонецЦикла;
КонецПроцедуры  // ПриЗаписи()
```

2. `НайтиПоКоду()`.

Метод `НайтиПоКоду(Код,ФлагПоиска)` находит элемент справочника по коду. Метод выполняет поиск элемента справочника по заданному коду и позиционирует объект на этом элементе.

Возвращает значения:

- 1 — если действие выполнено;
- 0 — если действие не выполнено (элемент не найден).

Параметры:

- `Код` — выражение со значением искомого кода;
- `ФлагПоиска` — необязательный параметр, который может принимать значения:
  - 0 — поиск во всем справочнике, вне зависимости от родителя;
  - 1 — поиск внутри установленного уровня подчинения (родителя);
  - 2 — поиск по полному коду через разделитель.

Для параметра ФлагПоиска определены значения по умолчанию:

- 0 — если код уникален во всем справочнике;
- 2 — если код уникален только в группе.

Для демонстрации примера, использующего этот метод, приведенную выше процедуру модифицируем следующим образом:

```
Процедура ПриЗаписи()
    Если Услуга.Выбран()=0 тогда
      Сообщить("Не выбран вид расчета - услуга! Документ не записан");
      Предупреждение("Не выбран вид расчета - услуга! Документ не
записан",30);
      СтатусВозврата(0);
      Возврат;
    КонецЕсли;
    Кв = СоздатьОбъект("Справочник.Квартиры");
    ВыбратьСтроки();
    Пока ПолучитьСтроку()=1 цикл
      Если Кв.НайтиПоКоду(Квартира.Код,0) = 1 тогда
        Кв.ПоказаниеХолВоды.Установить(КонМесяца(ДатаДок),
                                       ПоказаниеСчетчика);
        Кв.Записать();
      КонецЕсли;
    КонецЦикла;
КонецПроцедуры // ПриЗаписи()
```

3. НайтиПоНаименованию().

Метод НайтиПоНаименованию(Наименование,Режим,ФлагПоиска) находит элемент справочника по наименованию. Метод выполняет поиск элемента справочника по заданному наименованию и позиционирует объект на этом элементе.

Возвращает:

- 1 — если действие выполнено;
- 0 — если действие не выполнено (элемент не найден).

Параметры:

- Наименование — строка с наименованием искомого элемента справочника;
- Режим — число (необязательный параметр), принимающее значения:
  - 1 — поиск внутри установленного подчинения (родителя);
  - 0 — поиск во всем справочнике вне зависимости от родителя (значение по умолчанию — 1);

- ❒ `ФлагПоиска` — число (необязательный параметр), принимающее значения:
  - 1 — найти точное соответствие наименования;
  - 0 — найти наименование по первым символам (значение по умолчанию — 0).

Определим в форме элемента справочника "Контрагенты" предопределенную процедуру `ПриЗаписи()`. При этом зададим в ней контроль на дублирование наименования:

```
Процедура ПриЗаписи()
    Контр = СоздатьОбъект("Справочник.Контрагенты");
// Проверим, есть ли уже элемент с заданным наименованием при вводе
// нового элемента или при изменении наименования уже существующего
// элемента
    Если Контр.НайтиПоНаименованию(Наименование) = 1 тогда
      Если Вопрос("Элемент с наименованием "+Наименование +"
                  уже существует! Записать элемент?",4,30)=7 тогда
        СтатусВозврата(0);
      КонецЕсли;
    КонецЕсли;
КонецПроцедуры  // ПриЗаписи()
```

4. `НайтиПореквизиту()`.

Метод `НайтиПоРеквизиту(ИмяРеквизита,Значение,ФлагГлобальногоПоиска)` находит элемент справочника по значению реквизита. Метод выполняет поиск элемента справочника по заданному значению реквизита и позиционирует объект на этом элементе.

Возвращает значения:

- ❒ 1 — если действие выполнено;
- ❒ 0 — если действие не выполнено (элемент не найден).

Параметры:

- ❒ `ИмяРеквизита` — строка с наименованием реквизита;
- ❒ `Значение` — значение реквизита для поиска;
- ❒ `ФлагГлобальногоПоиска` — число, принимающее значения:
  - 0 — поиск выполняется в пределах подчинения справочника;
  - 1 — поиск выполняется по всему справочнику.

Внимание

Метод можно использовать только для реквизитов с установленным признаком **Сортировка**.

Добавим в приведенной ранее предопределенной процедуре `ПриЗаписи()` фрагмент с контролем по реквизиту **ИНН**.

```
Процедура ПриЗаписи()
   Контр = СоздатьОбъект("Справочник.Контрагенты");
   Если Контр.НайтиПоНаименованию(Наименование) = 1 тогда
     Если Вопрос("Элемент с наименованием "+Наименование +"
                 уже существует! Записать элемент?",4,30)=7 тогда
        СтатусВозврата(0);
     КонецЕсли;
   КонецЕсли;
   Если ПустаяСтрока(ИНН)=0 Тогда
// Проверим, есть ли элемент справочника с заданным значением
// реквизита ИНН
      Если Контр.НайтиПоРеквизиту("ИНН",ИНН,1) = 1 тогда
        Если Вопрос("Элемент с ИНН "+ИНН +" уже существует!
                    Записать элемент?",4,30)=7 тогда
           СтатусВозврата(0);
        КонецЕсли;
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

Для реквизита **ИНН** определен признак **Сортировка** (рис. 3.17).

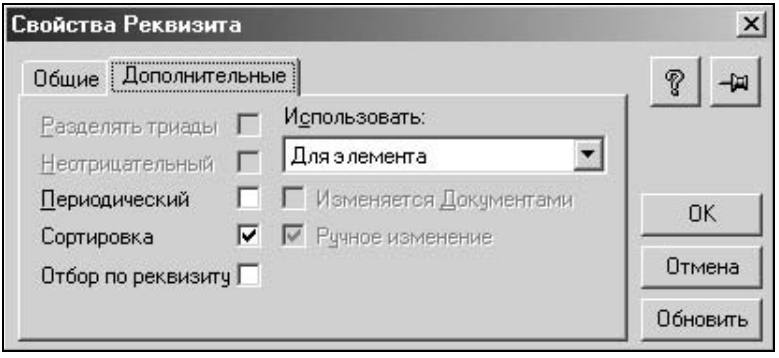


Рис. 3.17. Диалоговое окно свойств реквизита **ИНН** на вкладке **Дополнительные**

5. `ВыбратьЭлементы()`.

Метод `ВыбратьЭлементы(Режим)` открывает выборку элементов справочника.

Возвращает значения:

- 1 — если действие выполнено и в выборке есть хотя бы один элемент;
- 0 — если действие не выполнено или в выборке нет ни одного элемента.

Параметры:

- ❒ `Режим` — число, принимающее значения:
  - 1 — выбирать элементы с учетом иерархии;
  - 0 — выбирать элементы без учета иерархии (параметр необязателен, по умолчанию устанавливается значение 1).

В модуле формы справочника "Квартиры" была задана функция определения владельца квартиры для заполнения графы формы справочника **Владелец**. Рассмотрим использование метода на ее примере.

```
Функция ОпределениеВладельца()
  Жильцы = СоздатьОбъект("Справочник.Жильцы");
  Жильцы.ИспользоватьВладельца(ТекущийЭлемент());
// Откроем выборку элементов
  Жильцы.ВыбратьЭлементы();
  Пока Жильцы.ПолучитьЭлемент()=1 цикл
    Если Жильцы.СтатусПрописки = Перечисление.Статусы.Владелец тогда
      Возврат Жильцы.ФИО
    КонецЕсли;
  КонецЦикла;
КонецФункции  // ОпределениеВладельца
```

6. `ВыбратьЭлементыПоРеквизиту()`.

Метод `ВыбратьЭлементыПоРеквизиту(ИмяРеквизита,Значение, РежимИерархии, РежимГрупп)` открывает выборку элементов справочника по значению реквизита.

Возвращает значения:

- ❒ 1 — если действие выполнено и в выборке есть хотя бы один элемент;
- ❒ 0 — если действие не выполнено или в выборке нет ни одного элемента.

Параметры:

- ❒ `ИмяРеквизита` — строка с именем реквизита, по которому выполняется выборка;
- ❒ `Значение` — значение реквизита для выборки;
- ❒ `РежимИерархии` — число, принимающее значения:
  - 1 — выбирать элементы с учетом иерархии;
  - 0 — выбирать элементы без учета иерархии (параметр необязателен, по умолчанию устанавливается значение 1);
- ❒ `РежимГрупп` — число, принимающее значения:
  - 1 — выбирать среди групп справочника;
  - 0 — выбирать только среди элементов справочника.

Внимание

Метод можно использовать только для реквизитов с установленным признаком **Сортировка**.

Данный метод удобно использовать для создания предопределенных элементов справочников в случае их отсутствия. Для примера приведем функцию модуля документа "Выписка", в которой создается элемент справочника "Прочие доходы и расходы" с наименованием "Купля-продажа иностранной валюты" и видом прочих расходов и доходов — КупляПродажаИностраннойВалюты.

```
Функция ПолучитьЭлементПрочихДоходовРасходов()
  СпрПрочиеДР = СоздатьОбъект( "Справочник.ПрочиеДоходыИРасходы");
  СпрПрочиеДР.ВыбратьЭлементыПоРеквизиту
                                    ("ВидПрочихДоходовИРасходов",
  Перечисление.ВидыПрочихДоходовИРасходов.КупляПродажаИностраннойВалюты,,);
  Если СпрПрочиеДоходыИРасходы.ПолучитьЭлемент() = 1 Тогда
    КупляПродажаВалюты = СпрПрочиеДоходыИРасходы.ТекущийЭлемент();
  Иначе
    СпрПрочиеДоходыИРасходы.Новый();
    СпрПрочиеДоходыИРасходы.Наименование = "Купля-продажа
                                           иностранной валюты";
    СпрПрочиеДоходыИРасходы.ВидПрочихДоходовИРасходов = Перечисление.
    ВидыПрочихДоходовИРасходов.КупляПродажаИностраннойВалюты;
    СпрПрочиеДоходыИРасходы.Записать();
    КупляПродажаВалюты = СпрПрочиеДоходыИРасходы.ТекущийЭлемент();
  КонецЕсли;
  Возврат КупляПродажаВалюты;
КонецФункции  // ПолучитьЭлементПрочихДоходовРасходов()
```

7. ПолучитьЭлемент().

Метод ПолучитьЭлемент(Режим) получает из выборки следующий элемент справочника.

Возвращает значения:

- 1 — если элемент выбран;
- 0 — если элемент не выбран.

Параметр — Режим принимает значения:

- 1 — включать в выборку подчиненные элементы;
- 0 — не включать в выборку подчиненные элементы (параметр необязателен, по умолчанию устанавливается значение 1).

В приведенной выше функции ОпределениеВладельца() используется данный метод для просмотра жильцов квартиры и нахождения среди них владельца.

8. `ИспользоватьВладельца()`.

Метод `ИспользоватьВладельца(Владелец,ФлагИзменения)` позволяет установить выборку по элементу связанного справочника. Возвращает предыдущее значение текущего владельца для справочника (на момент до исполнения метода).

Параметры:

- `Владелец` — значение элемента связанного справочника, которому подчинен данный справочник;
- `ФлагИзменения` — флаг (необязательный параметр), регулирующий возможность интерактивного изменения владельца:
  - 1 — пользователь может изменить владельца интерактивно;
  - 0 — пользователь не может интерактивно изменить владельца (значение по умолчанию — 1).

В функции `ОпределениеВладельца()` используется данный метод для формирования выборки справочника "Жильцы", подчиненного справочнику "Квартиры":

```
Функция ОпределениеВладельца()
  Жильцы = СоздатьОбъект("Справочник.Жильцы");
// Определим выборку справочника жильцы по текущему элементу
// справочника "Квартиры"
  Жильцы.ИспользоватьВладельца(ТекущийЭлемент());
  Жильцы.ВыбратьЭлементы();
  Пока Жильцы.ПолучитьЭлемент()=1 цикл
    Если Жильцы.СтатусПрописки = Перечисление.Статусы.Владелец тогда
      Возврат Жильцы.ФИО
    КонецЕсли;
  КонецЦикла;
КонецФункции  // ОпределениеВладельца
```

9. `ИспользоватьРодителя()`.

Метод `ИспользоватьРодителя(Родитель,ФлагИзменения)` позволяет установить выборку элементов по группе справочника. Возвращает предыдущее значение текущей группы для справочника (на момент до исполнения метода).

Параметры:

- `Родитель` — значение группы справочника, среди элементов которой делается выборка;
- `ФлагИзменения` — число (необязательный параметр), которое может принимать значения:
  - 1 — пользователь может изменить родителя интерактивно;

- 0 — пользователь не может интерактивно изменить родителя (значение по умолчанию — 1).

10. `ВключатьПодчиненные()`.

Метод `ВключатьПодчиненные(Число)` позволяет установить флаг выборки всех подчиненных элементов. Возвращает текущее числовое значение режима выборки подчиненных элементов справочника (на момент до исполнения метода).

Параметр — `Число`, которое может принимать значения:

- 1 — включать в выборку подчиненные элементы;
- 0 — не включать в выборку подчиненные элементы (необязателен, по умолчанию принимает значение — 1).

Внимание

Метод должен вызываться раньше метода `ВыбратьЭлементы()`.

11. `ПорядокКодов()`.

Метод `ПорядокКодов()` позволяет установить порядок выборки элементов справочника по возрастанию кода элементов.

Внимание

Метод вызывается до вызова метода `ВыбратьЭлементы()`.

Данный метод используется для справочников, у которых на код возложена дополнительная функциональная нагрузка. Например, в справочнике "Налоги и отчисления" код означает номер предела:

```
Функция ПолучитьПервыйПредел(Реквизит)
  Если (Код = "ПФР_страх") или (Код = "ПФР_нак") Тогда
    Возврат "";
  КонецЕсли;
  Ставки = СоздатьОбъект("Справочник.СтавкиНалогов");
  Ставки.ИспользоватьВладельца(ТекущийЭлемент());
// Установим порядок кодов справочника в порядке возрастания
  Ставки.ПорядокКодов();
  Ставки.ВыбратьЭлементы();
// Нам нужен только первый предел, т.е. первый элемент в выборке
  Если Ставки.ПолучитьЭлемент() = 1 Тогда
    Если Реквизит = "Наименование" Тогда
      Возврат "Ставка для первого предела:";
```

```
        Иначе
          Возврат СокрЛП (Формат (Ставки.ПолучитьАтрибут(Реквизит).
                Получить
                (ИспользоватьДату()), "Ч15.2.,"));
        КонецЕсли;
      Иначе
        Возврат "";
      КонецЕсли;
КонецФункции
```

12. `ПорядокНаименований()`.

Метод `ПорядокНаименований()` позволяет установить порядок выборки элементов справочника по возрастанию наименования, т. е. по алфавиту.

Внимание

Метод вызывается до вызова метода `ВыбратьЭлементы()`.

13. `ПорядокРеквизита()`.

Метод `ПорядокРеквизита(ИмяРеквизита)` позволяет установить порядок выборки элементов справочника по возрастанию значения реквизита.

Параметр — `ИмяРеквизита` представляет строку с именем реквизита, по возрастанию значений которого производится выборка.

Внимание

Метод может использоваться только для реквизита с установленным признаком **Сортировка**. Метод вызывается до вызова метода `ВыбратьЭлементы()`.

14. `НоваяГруппа()`.

Метод `НоваяГруппа()` добавляет новую группу справочника.

Методы регистров

Для объекта типа "Регистры" определен атрибут `ИдентификаторРегистра`, посредством которого осуществляется доступ к регистру конфигурации конкретного вида.

Методы регистров делятся на две группы:

- методы работы с конкретными регистрами;
- методы работы с совокупностью регистров, которые называются объектами типа "Регистры".

Возникает справедливый вопрос, почему только для объектов типа "Регистр" определен объект совокупности всех. Ответ кроется в самом названии

компоненты, для которой определен объект типа "Регистр" — "Оперативный учет". Бухгалтерские итоги рассчитываются поквартально, регистры же необходимо рассчитывать оперативно.

Методы объекта "Регистры"

1. `РассчитатьРегистрыНа()`.

Метод `РассчитатьРегистрыНа(ГраницаРасчета,ГрафаОтбора)` позволяет рассчитать все регистры с установленным флагом временного расчета на начало события.

Параметры:

- `ГраницаРасчета` — значение типа дата, документ или позиция;
- `ГрафаОтбора` — строковое выражение, принимающее значения:
  - идентификатор графы отбора;
  - * — предполагает автоматический выбор графы отбора;
  - пустая строка — предполагает не использовать графу отбора.

Параметр `ГрафаОтбора` — необязательный, он устанавливает использование определенной графы отбора. Если параметр не указан, то назначается автоматический выбор графы отбора.

2. `РассчитатьРегистрыПо()`.

Метод `РассчитатьРегистрыПо(ГраницаРасчета,ГрафаОтбора)` позволяет рассчитать все регистры с установленным флагом временного расчета на конец события.

Параметры:

- `ГраницаРасчета` — значение типа дата, документ или позиция;
- `ГрафаОтбора` — строковое выражение, принимающее значения:
  - идентификатор графы отбора;
  - * — предполагает автоматический выбор графы отбора;
  - пустая строка — предполагает не использовать графу отбора.

Параметр `ГрафаОтбора` — необязательный, он устанавливает использование определенной графы отбора. Если параметр не указан, то назначается автоматический выбор графы отбора.

3. `Актуальность()`.

Метод `Актуальность(ФлагАктуальности)` устанавливает флаг актуальности временного расчета. Возвращает текущее состояние флага актуальности временного расчета:

- 1 — временный расчет поддерживается в актуальном состоянии;
- 0 — не поддерживается.

Параметр — **ФлагАктуальности** является необязательным и может принимать значения:

- 1 — временный расчет поддерживать в актуальном состоянии;
- 0 — не поддерживать актуальность временного расчета.

Если параметр не задан, то метод просто возвращает текущий флаг актуальности, не меняя его.

> **Внимание**
>
> Данный метод можно использовать только в модуле проведения документа. Если флаг установлен, то все последующие движения регистров будут изменять итоги временного расчета, т. е. итоги регистров временного расчета будут все время (при проведении документа) находиться в актуальном состоянии.

Пример использования методов объекта типа "Регистры" рассматривается в следующем разделе.

Методы объекта "Регистр"

1. `ВыбратьДвиженияДокумента()`.

Метод `ВыбратьДвиженияДокумента(Документ)` выбирает все движения регистра по документу.

Возвращает:

- 1 — если действие выполнено и в выборке есть хотя бы один элемент;
- 0 — если действие не выполнено или в выборке нет ни одного элемента.

Параметр — `Документ`, определяет значение типа "Документ".

Введем в нашей конфигурации регистр для расчетов с поставщиками (см. рис. 3.18).

Для ведения взаиморасчетов зададим перечисление **КодыОпераций** (рис. 3.19).

Инициируем в конце глобального модуля переменную КО:

```
// Инициализация переменной перечисления кодов операций
КО = Перечисление.КодыОпераций;
```

Реквизит **КодОперации** имеет **Тип значения — Перечисление.КодыОпераций** (см. рис. 3.20).

Зададим в нашей конфигурации документ "Списание денежных средств", в котором нажатием кнопки вызывается процедура `РассчитатьСумму()`, которая рассчитывает сумму списания. Код этой процедуры приводится ниже.

```
Процедура РассчитатьСумму()
  Если (РасчДокумент.Выбран() = 0) Тогда
    Предупреждение("Укажите расчетный документ", 60);
```

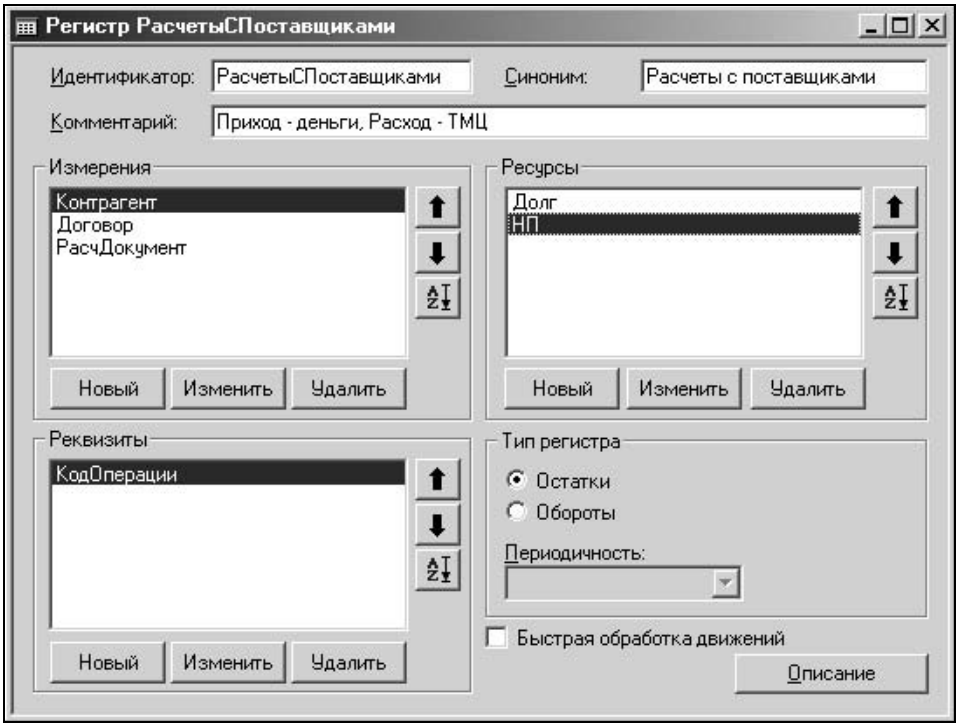


Рис. 3.18. Диалоговое окно **Регистр РасчетыСПоставщиками**

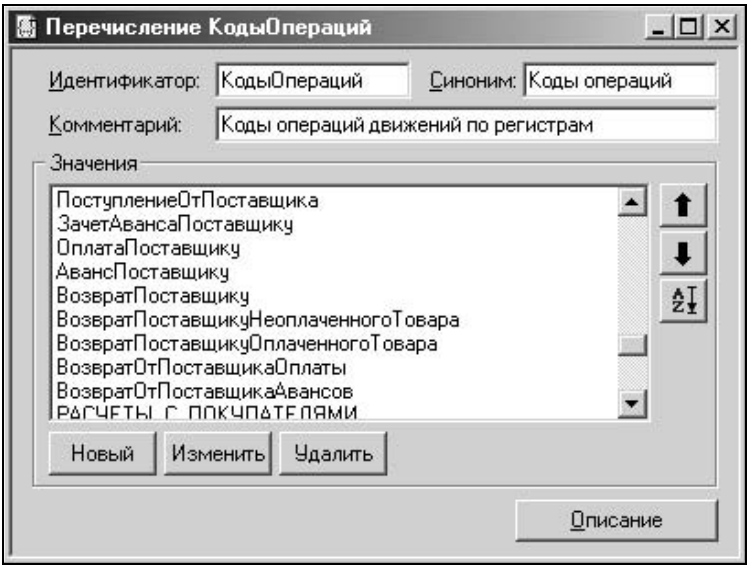


Рис. 3.19. Диалоговое окно **Перечисление КодыОпераций**

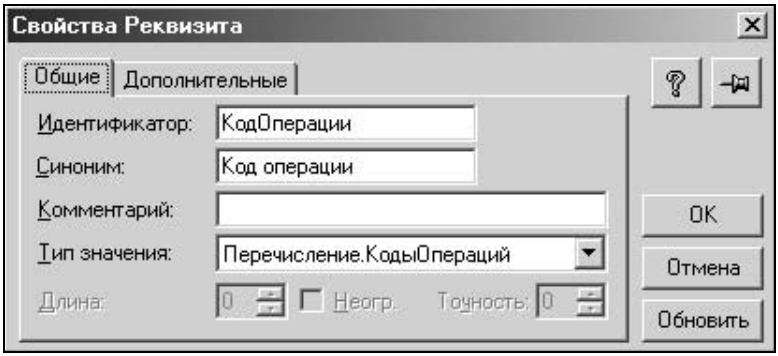


Рис. 3.20. Диалоговое окно свойств реквизита **КодОперации**

```
  Иначе
    Сумма = 0;
    СписокКодовПогашения = СоздатьОбъект("СписокЗначений");
    Если ВозвратОплата = 1 Тогда  // возврат
      РегРасчеты = СоздатьОбъект("Регистр.РасчетыСПокупателями");
// будем смотреть движения по кодам из списка
СписокКодовПогашения.ДобавитьЗначение(КО.ВозвратПокупателюОплаты);
СписокКодовПогашения.ДобавитьЗначение(КО.ВозвратПокупателюАвансов);
      ЗнакЗадолженности = 1;
    Иначе           // оплата
      РегРасчеты = СоздатьОбъект("Регистр.РасчетыСПоставщиками");
      СписокКодовПогашения.ДобавитьЗначение(КО.ОплатаПоставщику);
// кредиторская задолженность хранится в регистре со знаком /-/
      ЗнакЗадолженности = -1;
    КонецЕсли;
    РегРасчеты.УстановитьЗначениеФильтра
                    ("Контрагент", Контрагент, 1);
    РегРасчеты.УстановитьЗначениеФильтра
                    ("РасчДокумент", РасчДокумент, 1);
    РегРасчеты.ВыбратьИтоги();
    Пока РегРасчеты.ПолучитьИтог() = 1 Цикл
      Сумма = Сумма + ЗнакЗадолженности * РегРасчеты.Долг;
      Если (УчитыватьНП = 1) и (ВозвратОплата = 1) Тогда
        Сумма = Сумма + ЗнакЗадолженности * РегРасчеты.НП;
      КонецЕсли;
    КонецЦикла;
    Если (Проведен() = 1) Тогда
```

```
// Выберем движения документа
     РегРасчеты.ВыбратьДвиженияДокумента(ТекущийДокумент());
// Просмотрим все движения на предмет совпадения кода операции
// (метод ПолучитьДвижение() описан далее)
     Пока РегРасчеты.ПолучитьДвижение() = 1 Цикл
       Если СписокКодовПогашения.Принадлежит(РегРасчеты.КодОперации)
       = 1
          Тогда
         Сумма = Сумма + РегРасчеты.Долг;
         Если (УчитыватьНП = 1) и (ВозвратОплата = 1) Тогда
           Сумма = Сумма + РегРасчеты.НП;
         КонецЕсли;
       КонецЕсли;
     КонецЦикла;
   КонецЕсли;
 КонецЕсли;
КонецПроцедуры // РассчитатьСумму()
```

2. `ВыбратьДвижения()`.

Метод `ВыбратьДвижения(ДатаНачала,ДатаКонца,ГрафаОтбора)` выбирает все движения регистра по датам в заданном интервале дат.

Параметры:

- `ДатаНачала` — дата, документ или позиция начала временного интервала выбора движений регистра;
- `ДатаКонца` — дата, документ или позиция конца временного интервала выбора движений регистра (если не указана или 0, то конец — точка актуальности);
- `ГрафаОтбора` — необязательный параметр, строковое выражение в виде идентификатора графы отбора, определяющее использование определенной графы отбора:
  - идентификатор графы отбора;
  - * — автоматический выбор графы отбора;
  - пустая строка — не использовать графу отбора;
  - если не указан — автоматический выбор графы отбора.

Рассмотрим фрагмент отбора движений по регистру в документе "Счет-фактура полученный":

```
// Если разрешен зачет НДС, проведем это по регистрам
Если (ЗачетНДСприОплате = 1) Тогда
  ТаблицаЗачетаНДС = СоздатьОбъект ("ТаблицаЗначений");
```

```
ТаблицаЗачетаНДС.НоваяКолонка("Контрагент", "Справочник");
ТаблицаЗачетаНДС.НоваяКолонка("СчетФактура", "Документ");
ТаблицаЗачетаНДС.НоваяКолонка("ОплатаБазаНДС", "Число");
РегистрПоставщики = СоздатьОбъект("Регистр.РасчетыСПоставщиками");
// Установим фильтр по контрагенту
РегистрПоставщики.УстановитьЗначениеФильтра
                              ("Контрагент", Контрагент, 1);
// Выберем движения регистра по контрагенту
РегистрПоставщики.ВыбратьДвижения(ДокументОснование,
                              ТекущийДокумент(), "Контрагент");
// Просмотрим все движения на предмет совпадения документа-
// основания
Пока РегистрПоставщики.ПолучитьДвижение() = 1 Цикл
  Если (РегистрПоставщики.ТекущийДокумент() = ДокументОснование)
 Тогда
       Если (РегистрПоставщики.КодОперации =
    КО.ЗачетАвансаПоставщику)
          Тогда
      ТаблицаЗачетаНДС.НоваяСтрока();
      ТаблицаЗачетаНДС.Контрагент = РегистрПоставщики.Контрагент;
      ТаблицаЗачетаНДС.СчетФактура = ТекущийДокумент();
      ТаблицаЗачетаНДС.ОплатаБазаНДС = РегистрПоставщики.Долг -
                                       РегистрПоставщики.НП;
    КонецЕсли;
  ИначеЕсли (РегистрПоставщики.РасчДокумент = ДокументОснование)
        Тогда
    Если (РегистрПоставщики.Приход = 1) Тогда
      Если (РегистрПоставщики.КодОперации = КО.ОплатаПоставщику)
        Тогда
        ТаблицаЗачетаНДС.НоваяСтрока();
        ТаблицаЗачетаНДС.Контрагент =
                                  РегистрПоставщики.Контрагент;
        ТаблицаЗачетаНДС.СчетФактура = ТекущийДокумент();
        ТаблицаЗачетаНДС.ОплатаБазаНДС =
        РегистрПоставщики.Долг - РегистрПоставщики.НП;
      КонецЕсли;
    КонецЕсли;
  КонецЕсли;
КонецЦикла;
```

```
    ТаблицаЗачетаНДС.Свернуть
                        ("Контрагент, СчетФактура", "ОплатаБазаНДС");
    глЗачетНДС(Контекст, ТаблицаЗачетаНДС);
КонецЕсли;
```

3. `ПолучитьДвижение()`.

Метод `ПолучитьДвижение()` выбирает очередное движение регистра.

Возвращает значение:

- 1 — если следующее движение регистра выбрано;
- 0 — иначе.

Из приведенных выше примеров видно, что метод выбирает очередное движение из выборки независимо от того, каким методом выборка была получена: `ВыбратьДвиженияДокумента()` или `ВыбратьДвижения()`.

4. `ТекущийДокумент()`.

Метод `ТекущийДокумент()` возвращает документ, задавший движение регистра.

В вышеприведенном фрагменте кода используется данный метод:

```
Если (РегистрПоставщики.ТекущийДокумент() = ДокументОснование) Тогда
```

5. `НомерСтроки()`.

Метод `НомерСтроки()` возвращает номер строки документа, по которой было выбрано движение.

В стандартном отчете о движении документа используется данный метод:

```
СекцияДвижение.НомерСтроки = Рег.НомерСтроки();
```

6. `ВыбратьИтоги()`.

Метод `ВыбратьИтоги()` выбирает все остатки регистра.

В приведенной выше процедуре `РассчитатьСумму()` использовался данный метод:

```
    РегРасчеты.ВыбратьИтоги();
    Пока РегРасчеты.ПолучитьИтог() = 1 Цикл
```

7. `ПолучитьИтог()`.

Метод `ПолучитьИтог()` выбирает очередной остаток по регистру.

Возвращает:

- 1 — если очередной остаток по регистру выбран;
- 0 — иначе.

Внимание

Итоги с нулевыми остатками не выдаются. Порядок выдачи для измерений объектов типа "Справочник" и "Документ" не определен.

В приведенной выше процедуре `РассчитатьСумму()` использовался данный метод:

```
Пока РегРасчеты.ПолучитьИтог() = 1 Цикл
```

8. `ВременныйРасчет()`.

Метод `ВременныйРасчет(Флаг)` позволяет установить флаг участия регистра во временном расчете.

Возвращает текущее значение флага участия регистра во временном расчете.

Параметр — **Флаг** может принимать значения:

- 1 — установить флаг участия регистра во временном расчете;
- 0 — сбросить флаг участия регистра во временном расчете (необязателен, по умолчанию устанавливается значение — 1).

Внимание

В один момент только по одному объекту регистров каждого вида могут участвовать во временном расчете.

Введем в нашей конфигурации документ "Авансовый отчет" (см. рис. 3.21).

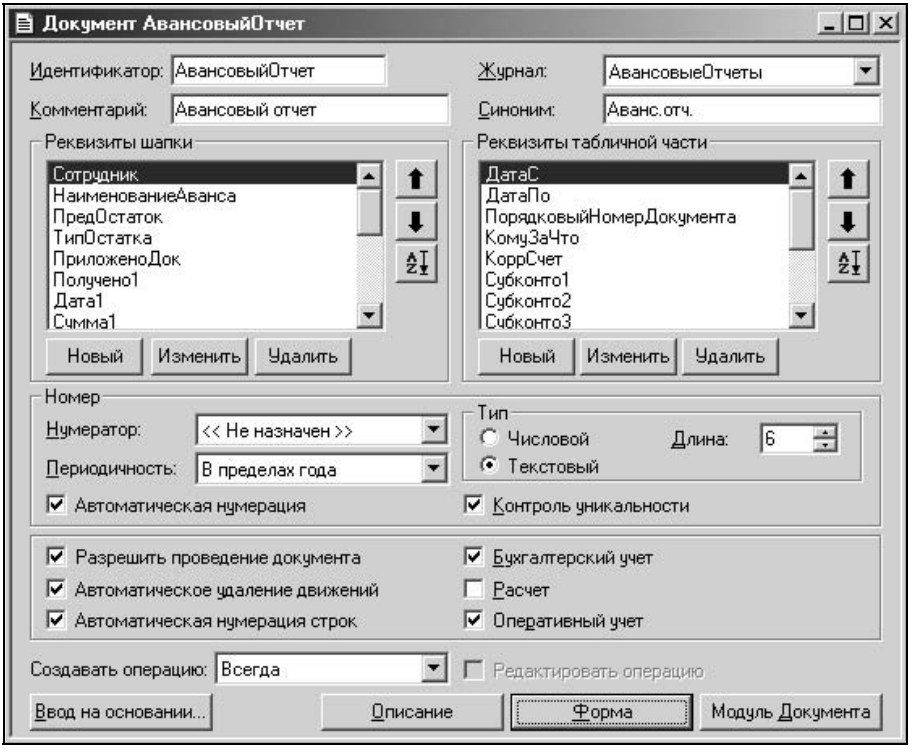


Рис. 3.21. Диалоговое окно **Документ АвансовыйОтчет**

Рассмотрим фрагмент предопределенной процедуры ОбработкаПроведения() модуля документа "Авансовый отчет":

```
Регистры = СоздатьОбъект("Регистры");
Расчеты = Регистры.РасчетыСПоставщиками;
СписокКонтрагентов = СоздатьОбъект("СписокЗначений");
ВыгрузитьТабличнуюЧасть(СписокКонтрагентов, "Субконто1");
Расчеты.УстановитьЗначениеФильтра
                              ("Контрагент", СписокКонтрагентов, 2);
СписокДоговоров = СоздатьОбъект("СписокЗначений");
ВыгрузитьТабличнуюЧасть(СписокДоговоров, "Субконто2");
Расчеты.УстановитьЗначениеФильтра("Договор", СписокДоговоров, 2);
Если (ИтогиАктуальны() = 0) Тогда
// Зададим участие нашего регистра во временном расчете
  Расчеты.ВременныйРасчет(1);
// Рассчитаем регистры на момент проведения нашего документа
  Регистры.Актуальность(1);
  Регистры.РассчитатьРегистрыПо(ТекущийДокумент());
КонецЕсли;
// В итоге для кода операции АвансПоставщику
Если (глСчетАвансовПоставщику(ДоговорПоставки) = КоррСчет)
                        и (СокрЛП(КоррСчет.Код) = "60.2")
Тогда
  Регистр.РасчетыСПоставщиками.Контрагент = Поставщик;
  Регистр.РасчетыСПоставщиками.Договор = ДоговорПоставки;
  Регистр.РасчетыСПоставщиками.РасчДокумент = ТекущийДокумент();
  Регистр.РасчетыСПоставщиками.Долг = Сумма;
  Регистр.РасчетыСПоставщиками.НП = 0;
  Регистр.РасчетыСПоставщиками.КодОперации = КО.АвансПоставщику;
  Регистр.РасчетыСПоставщиками.ДвижениеПриходВыполнить();
КонецЕсли;
```

Методы операций

1. ВыбратьОперации().

Метод ВыбратьОперации(НачалоПериода,КонецПериода) открывает выборку операций за период.

Возвращает значения:

- 1 — действие выполнено и в выборке есть хотя бы одна операция;
- 0 — действие не выполнено или в выборке нет ни одной операции.

Параметры:

- `НачалоПериода` — дата, документ или позиция начала периода выбора операций (необязательный параметр);
- `КонецПериода` — дата, документ или позиция конца периода выбора операций (необязательный параметр).

Определим в форме списка **Операция** кнопку **Печать** и процедуру для кнопки — `Печать()`:

```
Процедура Печать()
  Перем ИмяОтбора;
  Перем ЗначениеОтбора;
  Перем Док;
  Перем ВклПроводки;
  Док = 0;
  Меню=СоздатьОбъект("СписокЗначений");
  Меню.ДобавитьЗначение(0,"Не выводить проводки");
  Меню.ДобавитьЗначение(1,"Выводить проводки");
  Если Меню.ВыбратьЗначение(ВклПроводки,"",,, 1) = 0 Тогда
    Возврат;
  КонецЕсли;
//  Воспользуемся методом журнала ПолучитьОтбор
  ПолучитьОтбор(ИмяОтбора,ЗначениеОтбора);
  Дата1=НачалоИнтервала();
  Дата2=КонецИнтервала();
  Опер=СоздатьОбъект("Операция");
  Если Строка(ИмяОтбора)="" Тогда
// Если отбор не задан, выберем все операции за период, заданный в
// журнале
    Опер.ВыбратьОперации(Дата1,Дата2);
  ИначеЕсли (ИмяОтбора="Содержание") ИЛИ (ИмяОтбора="СуммаОперации")
  Тогда
// Если отбор задан по строке – Содержание
// или числу – СуммаОперации, выберем все операции по отбору
    Опер.ВыбратьПоЗначению(Дата1,Дата2,ИмяОтбора,ЗначениеОтбора);
  ИначеЕсли (ИмяОтбора="Контрагент") или (ИмяОтбора="Покупатель")
            или (ИмяОтбора="Продавец") или (ИмяОтбора="Договор") Тогда
// Если отбор задан по Контрагенту, выберем все документы по
// Контрагенту
    Док=СоздатьОбъект("Документ");
    Док.ВыбратьПоЗначению(Дата1,Дата2,ИмяОтбора,ЗначениеОтбора);
```

```
  Иначе
// В противном случае просто выберем все документы в заданном
// интервале
    Док=СоздатьОбъект("Документ."+ИмяОтбора);
    Док.ВыбратьДокументы(Дата1,Дата2);
  КонецЕсли;
  Таб=СоздатьОбъект("Таблица");
  Таб.ИсходнаяТаблица("ЖурналОпераций");
  Таб.ВывестиСекцию("Отчет");
  Таб.ВывестиСекцию(?(ВклПроводки=1,"ШапкаП","Шапка"));
  Таб.Опции(0,0,Таб.ВысотаТаблицы(),0);
  НПП=0;
  Если ТипЗначенияСтр(Док) <> "Документ" Тогда
// Переберем все операции последовательно
    Пока Опер.ПолучитьОперацию()=1 Цикл
      Состояние("Вывод "+Опер.ДатаОперации);
      НПП=НПП+1;
      Таб.ВывестиСекцию("Операция");
      Если ВклПроводки=1 Тогда
        Опер.ВыбратьПроводки();
        Пока Опер.ПолучитьПроводку()=1 Цикл
          Таб.ВывестиСекцию("Проводка");
        КонецЦикла;
      КонецЕсли;
    КонецЦикла;
  Иначе
    Пока Док.ПолучитьДокумент()=1 Цикл
      Если Док.СуществуетОперация()=1 Тогда
        Опер=Док.Операция;
        Состояние("Вывод "+Опер.ДатаОперации);
        НПП=НПП+1;
        Таб.ВывестиСекцию("Операция");
        Если ВклПроводки=1 Тогда
          Опер.ВыбратьПроводки();
          Пока Опер.ПолучитьПроводку()=1 Цикл
            Таб.ВывестиСекцию("Проводка");
          КонецЦикла;
        КонецЕсли;
```

```
        КонецЕсли;
      КонецЦикла;
  КонецЕсли;
  Таб.ВывестиСекцию("КонецОтчета");
  Таб.ТолькоПросмотр(1);
  Таб.Показать("Журнал операций","");
КонецПроцедуры
```

2. `ПолучитьОперацию()`.

Метод `ПолучитьОперацию()` выбирает очередную операцию из выборки, открытой при помощи метода `ВыбратьОперации()`.

Возвращает значения:

- 1 — операция выбрана успешно;
- 0 — операция не выбрана (отсутствует).

В приведенной выше процедуре `Печать()` использовался данный метод:

```
// Переберем все операции последовательно
```

Пока Опер.ПолучитьОперацию()=1 ***Цикл***

3. `НайтиОперацию()`.

Метод `НайтиОперацию(Документ)` осуществляет поиск операции по значению типа "Документ".

Возвращает значения:

- 1 — действие выполнено, операция найдена;
- 0 — действие не выполнено, операция не найдена.

Параметр — `Документ` определяется значением типа "Документ".

Атрибут Операция, для объекта типа "Документ", имеет смысл только для тех видов документов, для которых в конфигурации установлен признак **Бухгалтерский учет**. Атрибут и не используется как самостоятельное значение, а только позволяет обращаться к атрибутам и методам операции. Поэтому данный метод имеет широкое применение.

Рассмотрим фрагмент заполнения счета-фактуры на аванс на основании операции документа:

```
Опер = СоздатьОбъект("Операция");
Если ДокОсн.Вид() = "Выписка" Тогда
// Соберем проводки «Выписки» по полученным авансам в таблицу
// значений
  Авансы = СоздатьОбъект("ТаблицаЗначений");
  Авансы.НоваяКолонка("Контрагент", "Справочник",,, "Контрагент", 20);
  Авансы.НоваяКолонка("Договор", "Справочник",,, "Договор", 12);
```

```
  Авансы.НоваяКолонка("Сумма", "Число",,, "Сумма аванса", 10);
  Авансы.НоваяКолонка("СуммаВВалютеДоговора", "Число");
  Авансы.ВидимостьКолонки("СуммаВВалютеДоговора", 0);
  СчетАванса = ?(ДокОсн.БанковскийСчет.ТипСчета =
               Перечисление.ТипыБанковскихСчетов.Валютный,
               СчетПоКоду("62.22"), СчетПокоду("62.2"));
// Позиционируем указатель на операции документа-основания
  Опер.НайтиОперацию(ДокОсн);
  СписокПроводок =    "51,62.2;
                    |52,62.22;";
// Выберем проводки операции по заданному фильтру
  Опер.ВыбратьОперацииСПроводками(ДокОсн, ДокОсн, СписокПроводок);
  Пока Опер.ПолучитьПроводку() = 1 Цикл
     Авансы.НоваяСтрока();
     Если Опер.Дебет.Счет = СчетАванса Тогда
       Авансы.Контрагент = Опер.Дебет.Контрагенты;
       Авансы.Договор = Опер.Дебет.Договоры;
     Иначе
       Авансы.Контрагент = Опер.Кредит.Контрагенты;
       Авансы.Договор = Опер.Кредит.Договоры;
     КонецЕсли;
     Авансы.Сумма = ?(Опер.ВалСумма=0, Опер.Сумма, Опер.ВалСумма);
     Если (Опер.Валюта.Выбран() = 1) и
          (Авансы.Договор.ВалютаДоговора <> Опер.Валюта) Тогда
       Авансы.СуммаВВалютеДоговора = Опер.Сумма;
     Иначе
       Авансы.СуммаВВалютеДоговора = Авансы.Сумма;
     КонецЕсли;
  КонецЦикла;
  Если Авансы.КоличествоСтрок() = 0  Тогда
     Предупреждение("Документ "+глПредставлениеДокумента(ДокОсн)+"
                    |не содержит авансов.");
     Возврат 0;
// Если "Выпиской" проведено более одного аванса, предложим
// пользователю выбрать один их них.
  ИначеЕсли Авансы.КоличествоСтрок() = 1 Тогда
     Стр = 1;
  Иначе
     Стр = 0;
     Если Авансы.ВыбратьСтроку(Стр, "Выберите контрагента документа
          оплаты") = 0 Тогда
```

```
        Возврат 0;
      КонецЕсли;
      Если Стр = 0 Тогда
        Возврат 0;
      КонецЕсли;
    КонецЕсли;
// Введем значения реквизитов счета-фактуры на основании
// выбранного аванса
    УдалитьСтроки();
    НоваяСтрока();
    НазначитьТип("Товар", "Строка", 22);
    Товар = "Аванс (предв. оплата)";
    СтранаПроисхождения = "";
    ГТД = "";
    Авансы.ПолучитьСтрокуПоНомеру(Стр);
    Аванс = 1;
    СуммаНДСопределяетсяРасчетнымМетодом = 1;
    Контрагент = Авансы.Контрагент;
    Договор = Авансы.Договор;
    ПриВыбореДоговора();
    Всего = Авансы.СуммаВВалютеДоговора;
    СтавкаНДС = Константа.ОсновнаяСтавкаНДС;
    НДС = Окр Авансы.СуммаВВалютеДоговора *
          Константа.ОсновнаяСтавкаНДС.Ставка/
          (100+Константа.ОсновнаяСтавкаНДС.Ставка) ,2,1);
    Сумма = Всего;
    Цена = Всего;
КонецЕсли;
```

4. `Выбрана()`.

Метод `Выбрана()` определяет, позиционирован ли объект в настоящий момент на некоторой операции или нет.

Возвращает значения:

- 1 — операция выбрана;
- 0 — операция не выбрана.

Например, пусть для выполнения действия с операцией, в форме списка операции необходимо определить наличие хоть одной операции в журнале на заданный период. Это можно сделать следующим образом:

```
Если Выбрана()=0 Тогда
  Возврат;
КонецЕсли;
```

5. `ПланСчетов()`.

Метод `ПланСчетов()` выдает план счетов текущей проводки.

Возвращает значение типа "План Счетов".

6. `ВыбратьОперацииСПроводками()`.

Метод `ВыбратьОперацииСПроводками(НачалоПериода,КонецПериода,Фильтр,Валюта,ПланСчетов,РазделительУчета)` открывает выборку операций с проводками за указанный период. Метод может задаваться двумя способами. Если в условии отбора участвует несколько счетов, то параметр отбора проводок по счетам задается строкой фильтра, если же отбор проводок ведется по одному счету и одному корреспондирующему счету, то их можно задать непосредственными параметрами. В данном варианте основные условия отбора задаются специальной строкой-фильтром.

Возвращает значения:

- 1 — действие выполнено и в выборке есть хотя бы одна проводка;
- 0 — действие не выполнено или в выборке нет ни одной проводки.

Параметры:

- `НачалоПериода` — дата, документ или позиция начала периода выбора операций (необязательный параметр);
- `КонецПериода` — дата, документ или позиция конца периода выбора операций (необязательный параметр);
- `Фильтр` — условие отбора проводок (необязательный параметр), которое определяет критерии отбора проводок для включения последних в выборку;

> **Примечание**
>
> Если параметр не заполнен, в выборку включаются все проводки. В качестве значения данного параметра можно также передавать строку, в которой могут находиться одна или несколько корреспонденций счетов или символьных строк, разделяемых символом — точка с запятой.

- `Валюта` — значание типа "Справочник" (используемого для валютного учета), которое определяет признак отбора проводок по валюте (необязательный параметр);
- `ПланСчетов` — значение типа "План Счетов" (если параметр не указан, то действие происходит по всем планам счетов);
- `РазделительУчета` — значение разделителя учета (если параметр не указан, то действие происходит по всем значениям разделителя учета).

В приведенном выше фрагменте используется данный метод:

```
// Выберем проводки операции по заданному фильтру
  СписокПроводок =     "51,62.2;
```

```
                    |52,62.22;";
Опер.ВыбратьОперацииСПроводками(ДокОсн, ДокОсн, СписокПроводок);
```

7. `ВыбратьОперацииСПроводками()`.

Метод `ВыбратьОперацииСПроводками(НачалоПериода, КонецПериода, Счет, КорСчет, Флаг, Валюта, ПланСчетов, РазделительУчета)` открывает выборку операций с проводками за указанный период. В данном варианте основные условия отбора задаются указанием счета и корреспондирующего счета.

Возвращает значения:

- 1 — действие выполнено и в выборке есть хотя бы одна проводка;
- 0 — действие не выполнено или в выборке нет ни одной проводки.

Параметры:

- `НачалоПериода` — дата, документ или позиция начала периода выбора операций. Необязательный параметр;
- `КонецПериода` — дата, документ или позиция конца периода выбора операций (необязательный параметр);
- `Счет` — счет, по которому будут отбираться проводки;
- `КорСчет` — корреспондирующий счет, по которому будут отбираться проводки (параметр имеет смысл, если указан параметр `Счет`);
- `Флаг` — признак вида оборота, принимающий следующие значения:
  - 1 — отбирать проводки только по дебету счета;
  - 2 — отбирать проводки только по кредиту счета;
  - 3 — отбирать проводки и по дебету, и по кредиту (по умолчанию устанавливается значение — 3).
- `Валюта` — значение типа "Справочник" (вида справочника, используемого для валютного учета), определяющее признак отбора проводок по валюте (необязательный параметр);
- `ПланСчетов` — значение типа "ПланСчетов" (если параметр не указан, то по всем планам счетов);
- `РазделительУчета` — значение разделителя учета (если параметр не указан, то анализ ведется по всем значениям разделителя учета).

8. `ИспользоватьСубконто()`.

Метод `ИспользоватьСубконто(ВидСубконто,Субконто)` задает фильтр по субконто для функции `ВыбратьОперацииСПроводками()`. Метод может вызываться последовательно несколько раз. В этом случае фильтры, устанавливаемые этой функцией, суммируются.

Параметры:

- ВидСубконто — значение типа "ВидСубконто" (отбор проводок будет выполнен только для субконто указанного вида);
- Субконто — значение субконто (отбор проводок будет выполнен только для указанного субконто, кроме того, в качестве значения данного параметра можно передавать "Список значений").

В отчете "Акт сверки" определен фрагмент, в котором используется данный метод:

```
Опер = СоздатьОбъект("Операция");
Опер.ИспользоватьСубконто(ВидыСубконто.Контрагенты, Контрагент);
Если ПустоеЗначение(Договор) = 0 Тогда
  Опер.ИспользоватьСубконто(ВидыСубконто.Договоры, Договор);
КонецЕсли;
Опер.ВыбратьОперацииСПроводками(НачДата, КонДата, Фильтр);
Пока Опер.ПолучитьПроводку() = 1 Цикл
  КонецЦикла;
```

9. ИспользоватьКорСубконто().

Метод ИспользоватьКорСубконто(ВидСубконто,Субконто) задает фильтр по корреспондирующим субконто для функции ВыбратьОперацииСПроводками(). Метод может вызываться последовательно несколько раз. В этом случае фильтры, устанавливаемые этой функцией, суммируются.

Параметры:

- ВидСубконто — значение типа "ВидСубконто", опредляющее отбор проводок только для корреспондирующих субконто указанного вида;
- Субконто — значение субконто, определяющее отбор проводок только для указанного корреспондирующего субконто (кроме того, в качестве значения данного параметра можно передавать "Список значений").

Данный метод действует аналогично методу ИспользоватьСубконто(), только работает с корреспондирующим субконто.

10. ВыбратьПоЗначению().

Метод ВыбратьПоЗначению(НачалоПериода,КонецПериода,ВидОтбора,ЗначениеОтбора) открывает выборку операций или проводок, отобранных по значению отбора.

Возвращает значения:

- 1 — действие выполнено и в выборке есть хотя бы одна операция или проводка;
- 0 — действие не выполнено или в выборке нет ни одной операции или проводки.

Параметры:

- `НачалоПериода` — дата, документ или позиция начала периода выбора операций (необязательный параметр);
- `КонецПериода` — дата, документ или позиция конца периода выбора операций (необязательный параметр);
- `ВидОтбора` — символьная строка, определяющая название вида отбора;
- `ЗначениеОтбора` — значение отбора вида, указанного в параметре `ВидОтбора`.

В процедуре `Печать()` использовался данный метод:

```
//  Воспользуемся методом журнала ПолучитьОтбор
  ПолучитьОтбор(ИмяОтбора,ЗначениеОтбора);
  Дата1=НачалоИнтервала();
  Дата2=КонецИнтервала();
  Опер=СоздатьОбъект("Операция");
  Если Строка(ИмяОтбора)="" Тогда
//  Если отбор не задан, выберем все операции за период, заданный в
// журнале
    Опер.ВыбратьОперации(Дата1,Дата2);
  ИначеЕсли (ИмяОтбора="Содержание") ИЛИ (ИмяОтбора="СуммаОперации")
  Тогда
// Если отбор задан по строке - Содержание
// или числу - СуммаОперации, выберем все операции по отбору
    Опер.ВыбратьПоЗначению(Дата1,Дата2,ИмяОтбора,ЗначениеОтбора);
  ИначеЕсли (ИмяОтбора="Контрагент") или (ИмяОтбора="Покупатель")
          или (ИмяОтбора="Продавец") или (ИмяОтбора="Договор")
      Тогда
// Если отбор задан по Контрагенту, выберем все документы
// по Контрагенту
    Док=СоздатьОбъект("Документ");
    Док.ВыбратьПоЗначению(Дата1,Дата2,ИмяОтбора,ЗначениеОтбора);
  Иначе
// В противном случае просто выберем все документы в заданном
// интервале
    Док=СоздатьОбъект("Документ."+ИмяОтбора);
    Док.ВыбратьДокументы(Дата1,Дата2);
  КонецЕсли;
```

Методы объектов типа "БухгалтерскиеИтоги"

Получать бухгалтерские итоги можно по нескольким схемам.

- Основные итоги — схема получения бухгалтерских итогов с детализацией по субсчетам и субконто за фиксированный период.
- Итоги по запросу — схема получения выборки бухгалтерских итогов с детализацией по субсчетам и субконто вплоть до проводки с использованием режима запроса.

Для выбора схемы необходимо четко представлять, что по первой схеме:

- Единица измерения задаваемого периода — месяц.
- Итоги невозможно получить в корреспонденции с другими счетами.
- Невозможно получить детализацию в разрезе документа, операции и проводки.

Данные ограничения связаны с использованием файлов бухгалтерских итогов 1sbkttl.dbf (остатки) и 1sbkttlc.dbf (итоги). На практике, однако, данная схема используется ввиду своей простоты и эффективности в тех случаях, когда доступной в схеме детализации достаточно. Для методов схемы получения бухгалтерских итогов по запросу используются файлы операций 1soper.dbf, проводок 1sentry.dbf и отбора проводок по субконто ssbsel.dbf.

Для указанных схем получения бухгалтерских итогов определены общие методы.

Общие методы

1. `ИспользоватьПланСчетов()`.

Метод `ИспользоватьПланСчетов(ПланСчетов)` назначает план счетов, по которому будут выдаваться итоги.

Возвращает значение данной установки до вызова метода.

Параметр — `ПланСчетов` определяет значение типа "План Счетов". Если параметр не задан, то установка не меняется.

2. `ИспользоватьРазделительУчета()`.

Метод `ИспользоватьРазделительУчета(РазделительУчета)` позволяет установить значение разделителя учета.

Возвращает: значение данной установки до вызова метода.

Параметр — `РазделительУчета` определяется значением разделителя учета. Если не задан параметр, то установка не меняется.

3. `ОсновныеИтоги()`.

Метод `ОсновныеИтоги()` переводит объект в режим работы с основными итогами. Вызов этого метода имеет смысл тогда, когда ранее был выполнен

расчет временных итогов или запрос, но нужно вернуть объект к работе с основными итогами. При этом результаты запроса или расчета временных итогов теряются.

Методы основных итогов

Рассмотрим схему получения основных бухгалтерских итогов.

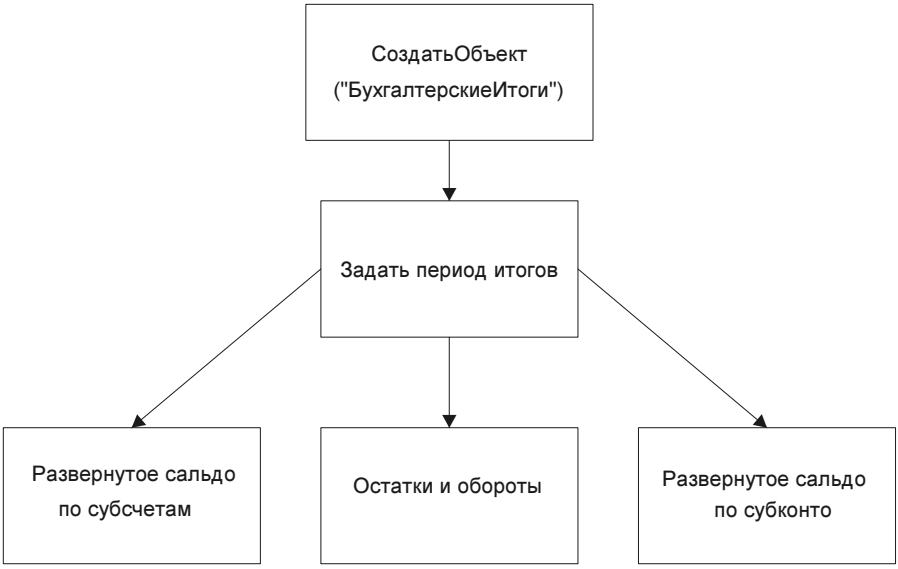


Рис. 3.22. Алгоритм получения основных бухгалтерских итогов

Как видно из схемы алгоритма (рис. 3.22), для получения основных бухгалтерских итогов вначале необходимо задать период итогов одним из методов, которые рассматриваются в *разд. "Период итогов"*. Дальнейшее применение методов является их комбинацией, которая рассматривается в *разд. "Итоги"*, *"Развернутое сальдо по субсчетам"* и *"Развернутое сальдо по субконто"*.

Период итогов

1. `ПериодД()`.

Метод `ПериодД(ДатаНачалаПериода,ДатаКонцаПериода)` устанавливает в качестве периода расчета итогов произвольное число месяцев.

Параметры:

- `ДатаНачалаПериода` — начальная дата периода выдачи итогов (дата должна быть равна дате начала месяца, необязательный параметр);
- `ДатаКонцаПериода` — конечная дата периода выдачи итогов (дата должна быть равна дате конца месяца, необязательный параметр).

2. ПериодКВ().

Метод ПериодКВ(Дата|НомерКвартала,Год) устанавливает квартал в качестве периода расчета итогов (где символ | определяет форму описания альтернативного значения).

Параметры:

- Дата — может принимать значения:
  - любая дата из квартала, устанавливаемого в качестве периода расчета итогов;
  - НомерКвартала — число от 1 до 4 (порядковый номер квартала);
- Год — год, заданный числом, включая век (используется только, если первый параметр — номер квартала).

3. ПериодКВН().

Метод ПериодКВН(Дата|НомерКвартала,Год) устанавливает в качестве периода расчета итогов период с начала года и до конца указанного квартала.

Параметры:

- Дата может принимать значения:
  - любая дата из квартала, устанавливаемого в качестве периода расчета итогов;
  - НомерКвартала число от 1 до 4 (порядковый номер квартала);
- Год — год, заданный числом, включая век (используется только, если первый параметр — номер квартала).

4. ПериодМ().

Метод ПериодМ(Дата|НомерМесяца,Год) устанавливает в качестве периода расчета итогов месяц.

Параметры:

- Дата может принимать значения:
  - любая дата из месяца, устанавливаемого в качестве периода расчета итогов;
  - НомерМесяца — число от 1 до 12 (порядковый номер месяца);
- Год — год, заданный числом, включая век (используется только, если первый параметр — номер месяца).

5. ПериодМНК().

Метод ПериодМНК(Дата|НомерМесяца,Год) устанавливает в качестве периода расчета итогов период с начала квартала до конца указанного месяца.

Параметры:

- ❒ `Дата` — может принимать значения:
  - любая дата из квартала, устанавливаемого в качестве периода расчета итогов;
  - `НомерМесяца` — число от 1 до 12 (порядковый номер месяца);
- ❒ `Год` — год, заданный числом, включая век (используется только, если первый параметр — номер месяца).

6. `ПериодМНГ()`.

Метод `ПериодМНГ(Дата|НомерМесяца,Год)` устанавливает в качестве периода расчета итогов период с начала года до конца указанного месяца.

Параметры:

- ❒ `Дата` — может принимать значения:
  - любая дата из квартала, устанавливаемого в качестве периода расчета итогов;
  - **НомерМесяца** — число от 1 до 12 (порядковый номер месяца);
- ❒ `Год` — год, заданный числом, включая век (используется только, если первый параметр — номер месяца).

7. `НачПериода()`.

Метод `НачПериода()` возвращает начальную дату установленного в данный момент периода основных итогов.

8. `КонПериода()`.

Метод `НачПериода()` возвращает конечную дату установленного в данный момент периода основных итогов.

Итоги

1. `СНД()`.

Метод `СНД(Счет,ТипСуммы,Валюта,Субконто1...)` возвращает дебетовое сальдо по счету на начало периода.

Параметры:

- ❒ `Счет` — значение типа "Счет", представляющее счет расчета итогов (может использоваться строка — код счета);
- ❒ `ТипСуммы` — число или строка (необязательный параметр), представляющие тип возвращаемой суммы, которые могут принимать одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;

- 3 (К) — количество;
- если параметр не указан, метод возвращает сумму;

- Валюта — значение типа "Справочник.Валюты" (необязательный параметр), причем, если параметр не указан, итоги выдаются без учета валюты;
- Субконто1, Субконто2, Субконто3 — значения субконто (необязательные параметры), причем их количество зависит от настройки субконто для данного счета (если параметры не указаны, то итоги выдаются без учета аналитики).

2. СНК().

Метод СНК(Счет,ТипСуммы,Валюта,Субконто1...) возвращает кредитовое сальдо по счету на начало периода.

Параметры:

- Счет — значение типа "Счет", представляющее счет расчета итогов (может использоваться строка — код счета);
- ТипСуммы — число или строка (необязательный параметр), определяющие тип возвращаемой суммы и принимающие одно из следующих значений:
  - 1 (С) — сумма;
  - 2 (В) — валютная сумма;
  - 3 (К) — количество;
  - если параметр не указан, метод возвращает сумму;
- Валюта — значение типа "Справочник.Валюты" (если параметр не указан, то итоги выдаются без учета валюты);
- Субконто1, Субконто2, Субконто3 — значения субконто (необязательные параметры), причем их количество зависит от настройки субконто для данного счета (если параметры не указаны, то итоги выдаются без учета аналитики).

3. СКД().

Метод СКД(Счет,ТипСуммы,Валюта,Субконто1...) возвращает дебетовое сальдо по счету на конец периода.

Параметры:

- Счет — значение типа "Счет", представляющее счет расчета итогов (может использоваться строка — код счета);
- ТипСуммы — число или строка (необязательный параметр), определяющие тип возвращаемой суммы (может принимать одно из следующих значений):
  - 1 (С) — сумма;
  - 2 (В) — валютная сумма;

- 3 (К) — количество;
- если параметр не указан, метод возвращает сумму;

- ❒ `Валюта` — значение типа "Справочник.Валюты" (необязательный параметр), причем, если параметр не указан, то итоги выдаются без учета валюты;
- ❒ `Субконто1, Субконто2, Субконто3` — значения субконто (необязательные параметры), причем их количество зависит от настройки субконто для данного счета (если параметры не указаны, то итоги выдаются без учета аналитики).

Рассмотрим простой пример использования метода `СКД()` с количественным итогом. Создадим в нашей конфигурации справочник "Материалы" (см. рис. 3.23).

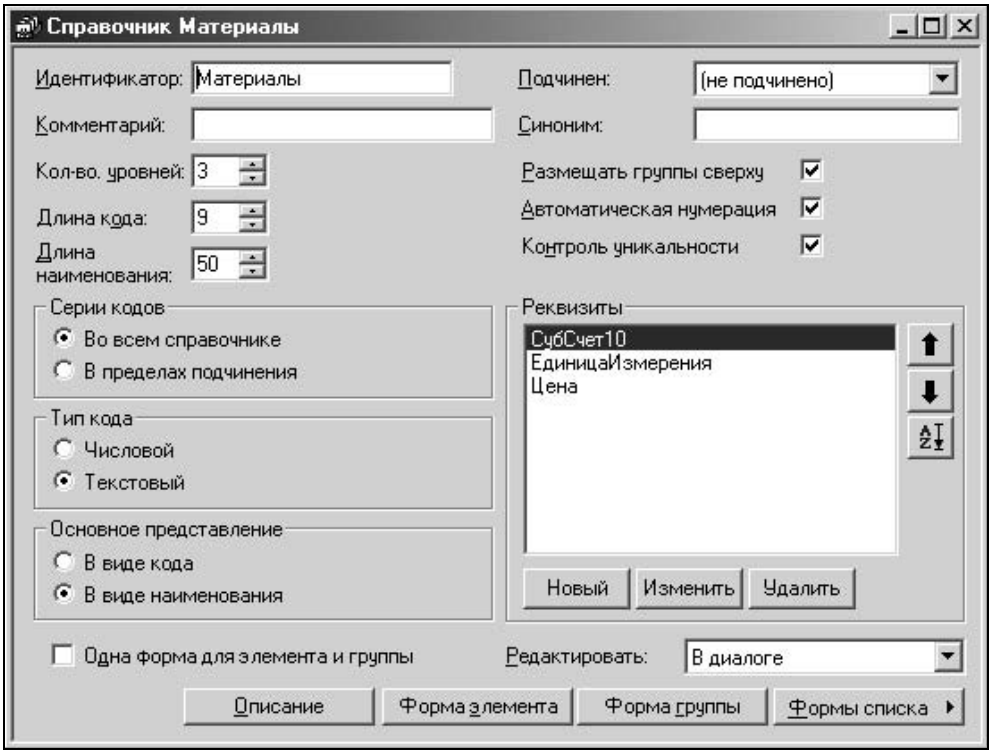


Рис. 3.23. Диалоговое окно справочника "Материалы"

Зададим для справочника форму списка, в которой **Остаток** — колонка (рис. 3.24), рассчитываемая функцией `ОстатокНаСкладе()`:

```
Функция ОстатокНаСкладе()
  Остаток = 0;
```

```
   Если (КонМесяца(КонтекстФормыДокумента.ДатаДок)
   <= КонецРассчитанногоПериодаБИ()) и (ТекущийЭлемент().Выбран() = 1)
   Тогда
      Если МестоХранения.Выбран() = 1 Тогда
// ТекущийЭлемент().СубСчет10 - счет
//  "К" - тип суммы
//  ТекущийЭлемент()  - Субконто1
//  МестоХранения - Субконто2
        Остаток = БухИт.СКД(ТекущийЭлемент().СубСчет10, "К",,
                            ТекущийЭлемент(),  МестоХранения);
      Иначе
        Остаток = БухИт.СКД(ТекущийЭлемент().СубСчет10, "К",,
                            ТекущийЭлемент());
      КонецЕсли;
   КонецЕсли;
   Возврат СокрЛ(Формат(Остаток, "Ч15.3."));
КонецФункции  // ОстатокНаСкладе
```

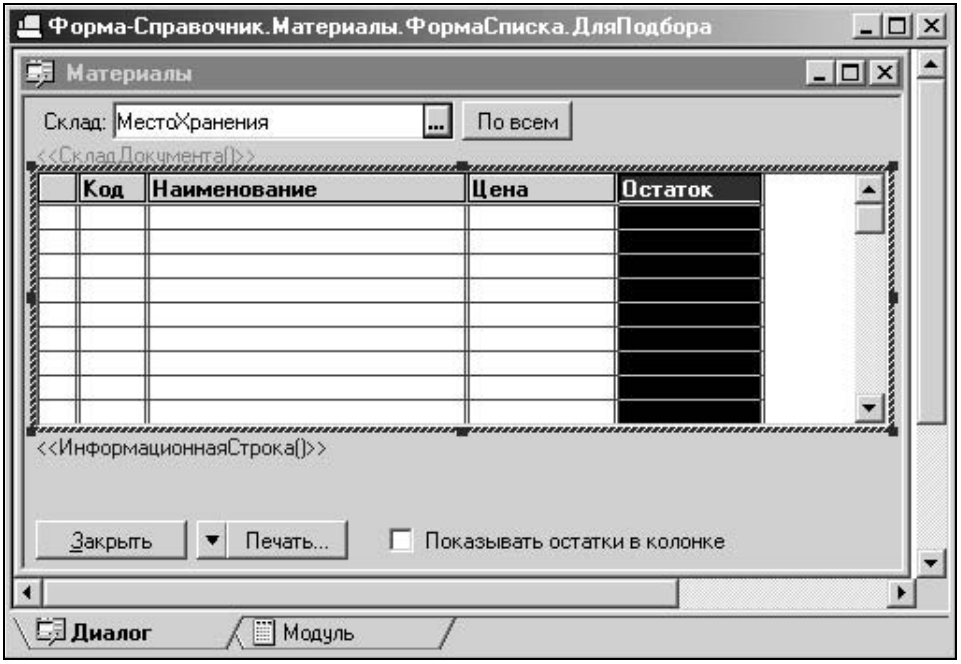


Рис. 3.24. Диалоговое окно формы справочника "Материалы"

В предопределенной процедуре `ПриОткрытии()` должен быть создан объект типа "БухгалтерскиеИтоги" и задан период итогов. Это можно сделать следующим образом:

```
Процедура ПриОткрытии()
  БухИт = СоздатьОбъект("БухгалтерскиеИтоги");
  БухИт.ПериодМ(РабочаяДата());
КонецПроцедуры  //ПриОткрытии
```

Чтобы переменная `БухИт` была доступна всем процедурам и функциям, определим ее в начале модуля:

```
Перем БухИт;
```

4. `СКК()`.

Метод `СКК(Счет,ТипСуммы,Валюта,Субконто1...)` возвращает кредитовое сальдо по счету на конец периода.

Параметры:

- `Счет` — значение типа "Счет", определяющее счет расчета итогов (может использоваться строка — код счета);
- `ТипСуммы` — число или строка (необязательный параметр), определяющие тип возвращаемой суммы и принимающие одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;
  - 3 (`К`) — количество;
  - если параметр не указан, метод возвращает сумму;
- `Валюта` — значение типа "Справочник.Валюты" (необязательный параметр), причем, если параметр не указан, то итоги выдаются без учета валюты;
- `Субконто1, Субконто2, Субконто3` — значения субконто (необязательные параметры), причем их количество зависит от настройки субконто для данного счета (если параметры не указаны, то итоги выдаются без учета аналитики).

5. `ДО()`.

Метод `ДО(Счет,ТипСуммы,Валюта,Субконто1...)` возвращает дебетовый оборот по счету на начало периода.

Параметры:

- `Счет` — значение типа "Счет", определяющее счет расчета итогов (может использоваться строка код счета);
- `ТипСуммы` — число или строка (необязательный параметр), определяющие тип возвращаемой суммы и принимающие одно из следующих значений:
  - 1 (`С`) — сумма;

- 2 (В) — валютная сумма;
- 3 (К) — количество;
- если параметр не указан, метод возвращает сумму;

❒ Валюта — значение типа "Справочник.Валюты" (необязательный параметр), причем, если параметр не указан, то итоги выдаются без учета валюты;

❒ Субконто1, Субконто2, Субконто3 — значения субконто (необязательные параметры), причем их количество зависит от настройки субконто для данного счета (если параметры не указаны, то итоги выдаются без учета аналитики).

6. КО().

Метод КО(Счет,ТипСуммы,Валюта,Субконто1...) возвращает кредитовый оборот по счету на начало периода.

Параметры:

❒ Счет — значение типа "Счет", определяющее счет расчета итогов (может использоваться строка — код счета);

❒ ТипСуммы — число или строка (необязательный параметр), определяющие тип возвращаемой суммы и принимающие одно из следующих значений:

- 1 (С) — сумма;
- 2 (В) — валютная сумма;
- 3 (К) — количество;
- если параметр не указан, метод возвращает сумму;

❒ Валюта — значение типа "Справочник.Валюты" (необязательный параметр), причем, если параметр не указан, итоги выдаются без учета валюты;

❒ Субконто1,Субконто2,Субконто3 — значения субконто (необязательные параметры), причем их количество зависит от настройки субконто для данного счета (если параметры не указаны, то итоги выдаются без учета аналитики).

7. ОБ().

Метод ОБ(СчетДеб,СчетКред,ТипСуммы,Валюта) позволяет рассчитать обороты между счетами. Возвращает число, представляющее оборот с дебета счета СчетДеб в кредит счета СчетКред.

Параметры:

❒ СчетДеб — значения типа "Счет", представляющие счет дебета, для которого необходимо выдать перекрестные обороты (может использоваться строка — код счета);

❒ СчетКред — значения типа "Счет", представляющие счета кредита, для которого необходимо выдать перекрестные обороты (может использоваться строка — код счета);

- `ТипСуммы` — число или строка (необязательный параметр), представляющие тип возвращаемой суммы, в виде:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;
  - 3 (`К`) — количество (если параметр не указан, метод возвращает сумму);
- `Валюта` — необязательный параметр, имеющий значение типа "Справочник.Валюты" (если параметр не указан, то итоги выдаются без учета валюты).

Развернутое сальдо по субсчетам

1. `СНДР()`.

Метод `СНДР(Счет,ТипСуммы,Валюта)` возвращает дебетовое развернутое сальдо по субсчетам на начало периода.

Параметры:

- `Счет` — значение типа "Счет", представляющее счет, для которого необходимо рассчитать развернутое сальдо (может использоваться строка — код счета);
- `ТипСуммы` — число или строка (необязательный параметр), определяет тип возвращаемой суммы и может принимать одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;
  - 3 (`К`) — количество;
  - если параметр не указан, метод возвращает сумму;
- `Валюта` — значение типа "Справочник.Валюты" (необязательный параметр), причем, если параметр не указан, то итоги выдаются без учета валюты.

2. `СНКР()`.

Метод `СНКР(Счет,ТипСуммы,Валюта)` возвращает кредитовое развернутое сальдо по субсчетам на начало периода.

Параметры:

- `Счет` — значение типа "Счет", определяющее счет, для которого необходимо рассчитать развернутое сальдо (может использоваться строка — код счета);
- `ТипСуммы` — число или строка (необязательный параметр), представляющий тип возвращаемой суммы, который может принимать одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;

- 3 (К) — количество;
- если параметр не указан, метод возвращает сумму;

❐ Валюта — значение типа "Справочник.Валюты" (если параметр не указан, то итоги выдаются без учета валюты).

3. СКДР().

Метод СКДР(Счет,ТипСуммы,Валюта) возвращает дебетовое развернутое сальдо по субсчетам на конец периода.

Параметры:

❐ Счет — значение типа "Счет", представляющее счет, для которого необходимо рассчитать развернутое сальдо (может использоваться строка — код счета);

❐ ТипСуммы — число или строка (необязательный параметр), представляющие тип возвращаемой суммы, который может принимать одно из следующих значений:

- 1 (С) — сумма;
- 2 (В) — валютная сумма;
- 3 (К) — количество;
- если параметр не указан, метод возвращает сумму;

❐ **Валюта** — значение типа "Справочник.Валюты" (необязательный параметр), причем, если параметр не указан, то итоги выдаются без учета валюты.

4. СККР().

Метод СККР(Счет,ТипСуммы,Валюта) возвращает кредитовое развернутое сальдо по субсчетам на конец периода.

Параметры:

❐ Счет — значение типа "Счет", представляющий счет, для которого необходимо рассчитать развернутое сальдо (может использоваться строка — код счета);

❐ ТипСуммы — число или строка (необязательный параметр), представляющие тип возвращаемой суммы, которое может принимать одно из следующих значений:

- 1 (С) — сумма;
- 2 (В) — валютная сумма;
- 3 (К) — количество;
- если параметр не указан, метод возвращает сумму;

❐ Валюта — значение типа "Справочник.Валюты" (необязательный параметр), причем, если параметр не указан, то итоги выдаются без учета валюты.

Развернутое сальдо по субконто

1. `СНДРС()`.

Метод `СНКРС(Счет,ТипСуммы,Валюта,Субконто1,ТипФильтра1,Субконто2, ТипФильтра2)` возвращает дебетовое развернутое сальдо по субконто на начало периода.

Параметры:

- `Счет` — значение типа "Счет", определяющее счет, для которого необходимо рассчитать развернутое сальдо (может использоваться строка — код счета);
- `ТипСуммы` — число или строка (необязательный параметр), представляющие тип возвращаемой суммы, который может принимать одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;
  - 3 (`К`) — количество;
  - если параметр не указан, метод возвращает сумму;
- `Валюта` — значение типа "Справочник.Валюты" (необязательный параметр), причем если параметр не указан, то итоги выдаются без учета валюты;
- `Субконто1, Субконто2, Субконто3` — значения субконто;
- `ТипФильтра1, ТипФильтра2, ТипФильтра3` — тип использования субконто, являющиеся строкой или числом, которые могут принимать значения:
  - * — разворачивать по данному субконто;
  - ! — отбирать по данному субконто;
  - " " — не учитывать данный субконто (значения по умолчанию — *, для первого субконто, и " " — для остальных субконто).

Зададим процедуру формирования на печать извещения на оплату коммунальных услуг:

```
Процедура Печать()
  Таб=СоздатьОбъект("Таблица");
  Таб.ИсходнаяТаблица("Таблица");
  Таб.ВывестиСекцию("Шапка");
  Влад = СоздатьОбъект("Справочник.Жильцы");
  Влад.ИспользоватьВладельца(Объект);
  Влад.ВыбратьЭлементы();
  Пока Влад.ПолучитьЭлемент()=1 цикл
    Если Влад.СтатусПрописки = Перечисление.Статусы.Владелец тогда
      Таб.Область("Плательщик").Текст = Влад.ФИО;
      Прервать;
```

```
   КонецЕсли;
КонецЦикла;
БИ = СоздатьОбъект("БухгалтерскиеИтоги");
// Зададим период бух. итогов на начало текущего периода
// журнала расчетов
БИ.ПериодМ(ДатаНачала,ДатаГод(ДатаНачала));
Сальдо=БИ.СНДРС(СчетПоКоду("62.1"),"С",,Влад.Контрагент,2);
Таб.Область("Долг").Текст = "Долг на "+ ДатаНачала + ":  "+ Сальдо;
ТаблицаТарифов = СоздатьОбъект("ТаблицаЗначений");
ТаблицаТарифов.НоваяКолонка("ВидРасчета","ВидРасчета");
ТаблицаТарифов.НоваяКолонка("Тариф");
Жильцы = СоздатьОбъект("ЖурналРасчетов.Начисления");
Тарифы = СоздатьОбъект("Справочник.Услуги");
Тарифы.ВыбратьЭлементы();
Пока Тарифы.ПолучитьЭлемент()=1 Цикл
   ТаблицаТарифов.НоваяСтрока();
   ТаблицаТарифов.ВидРасчета =  Тарифы.Расчет;
   ТаблицаТарифов.Тариф = Тарифы.Цена.Получить( Жильцы.
                          КонецТекущегоПериода());
КонецЦикла;
Расшифровка = СоздатьОбъект("ТаблицаЗначений");
Расшифровка.НоваяКолонка("ВидРасчета","ВидРасчета");
Расшифровка.НоваяКолонка("Тариф");
Расшифровка.НоваяКолонка("Начислено");
Расшифровка.НоваяКолонка("НачисленоЛ");
Жильцы.ВыбратьЗаписиПоОбъекту
                              (Объект,Жильцы.НачалоТекущегоПериода(),
                              Жильцы.КонецТекущегоПериода());
Итого = 0;
Пока Жильцы.ПолучитьЗапись()=1 цикл
    Расшифровка.НоваяСтрока();
    Расшифровка.ВидРасчета = Жильцы.ВидРасч;
    стр=0;
    кол=0;
    ТаблицаТарифов.НайтиЗначение(Жильцы.ВидРасч,стр,кол);
    ТаблицаТарифов.ПолучитьСтрокуПоНомеру(стр);
    Расшифровка.Тариф = ТаблицаТарифов.Тариф;
    Расшифровка.Начислено = Жильцы.Результат;
    Расшифровка.НачисленоЛ = Жильцы.Результат/ Жильцы.Объект.
    Льгота.Получить( Жильцы.КонецТекущегоПериода());
```

```
    Итого = Итого+Жильцы.Результат;
КонецЦикла;
Расшифровка.ВыбратьСтроки();
Таб.ВывестиСекцию("Итого");
Плательщик = Влад.ФИО;
Таб.ВывестиСекцию("Шапка2");
Пока Расшифровка.ПолучитьСтроку()=1 цикл
   ВидР = Расшифровка.ВидРасчета;
   Тариф = Расшифровка.Тариф;
   НачислениеПолное = Расшифровка.Начислено;
   НачислениеЛ = Расшифровка.НачисленоЛ;
   Таб.ВывестиСекцию("Строка");
КонецЦикла;
Таб.Опции(0,0,0,0,"ОпцииСправки");
Таб.ТолькоПросмотр(1);
Таб.Показать("Извещение","");
КонецПроцедуры
```

2. `СНКРС()`.

Метод `СНКРС(Счет,ТипСуммы,Валюта,Субконто1,ТипФильтра1,Субконто2, ТипФильтра2)` возвращает кредитовое развернутое сальдо по субконто на начало периода.

Параметры:

- `Счет` — значение типа "Счет", представляющее счет, для которого необходимо рассчитать развернутое сальдо (может использоваться строка — код счета);
- `ТипСуммы` — число или строка (необязательный параметр), представляющие тип возвращаемой суммы, который может принимать одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;
  - 3 (`К`) — количество;
  - если параметр не указан, метод возвращает сумму;
- `Валюта` — значение типа "Справочник.Валюты" (необязательный параметр), причем если параметр не указан, то итоги выдаются без учета валюты;
- `Субконто1, Субконто2,Субконто3` — значения субконто;
- `ТипФильтра1,ТипФильтра2,ТипФильтра3` — тип использования субконто, представлен строкой или числом, которые могут принимать значения:
  - * — разворачивать по данному субконто;
  - ! — отбирать по данному субконто;

- " " — не учитывать данный субконто (значение по умолчанию * — для первого субконто," " — для остальных субконто).

3. СКДРС().

Метод СКДРС(Счет,ТипСуммы,Валюта,Субконто1,ТипФильтра1,Субконто2, ТипФильтра2) возвращает дебетовое развернутое сальдо по субконто на конец периода.

Параметры:

- Счет — значение типа "Счет", представляющее счет, для которого необходимо рассчитать развернутое сальдо (может использоваться строка — код счета);
- ТипСуммы — число или строка (необязательный параметр), представляющие тип возвращаемой суммы, который может принимать одно из следующих значений:
  - 1 (С) — сумма;
  - 2 (В) — валютная сумма;
  - 3 (К) — количество;
  - если параметр не указан, метод возвращает сумму;
- Валюта — значение типа "Справочник.Валюты" (необязательный параметр), причем если параметр не указан, то итоги выдаются без учета валюты;
- Субконто1, Субконто2, Субконто3 — значения субконто;
- ТипФильтра1, ТипФильтра2, ТипФильтра3 — тип использования субконто (строка или число), представленные значением:
  - * — разворачивать по данному субконто;
  - ! — отбирать по данному субконто;
  - " " — не учитывать данный субконто (значение по умолчанию — *, для первого субконто и " " — для остальных субконто.

4. СККРС().

Метод СККРС(Счет,ТипСуммы,Валюта,Субконто1,ТипФильтра1,Субконто2, ТипФильтра2) возвращает кредитовое развернутое сальдо по субконто на конец периода.

Параметры:

- Счет — значение типа "Счет", определяющее счет, для которого необходимо рассчитать развернутое сальдо (может использоваться строка — код счета);
- ТипСуммы — число или строка (необязательный параметр), представляющее тип возвращаемой суммы, которое может принимать одно из следующих значений:
  - 1 (С) — сумма;
  - 2 (В) — валютная сумма;

- 3 (`К`) — количество;
- если параметр не указан, метод возвращает сумму;

- `Валюта` — значение типа "Справочник.Валюты" (необязательный параметр), причем если параметр не указан, то итоги выдаются без учета валюты;
- `Субконто1, Субконто2, Субконто3` — значения субконто;
- `ТипФильтра1, ТипФильтра2, ТипФильтра3` — тип использования субконто (строка или число), который может принимать значения:
  - * — разворачивать по данному субконто;
  - ! — отбирать по данному субконто;
  - " " — не учитывать данный субконто (значение по умолчанию — *, для первого субконто, " " — для остальных субконто).

Методы режима запроса

Рассмотрим схему получения бухгалтерских итогов с помощью запроса.

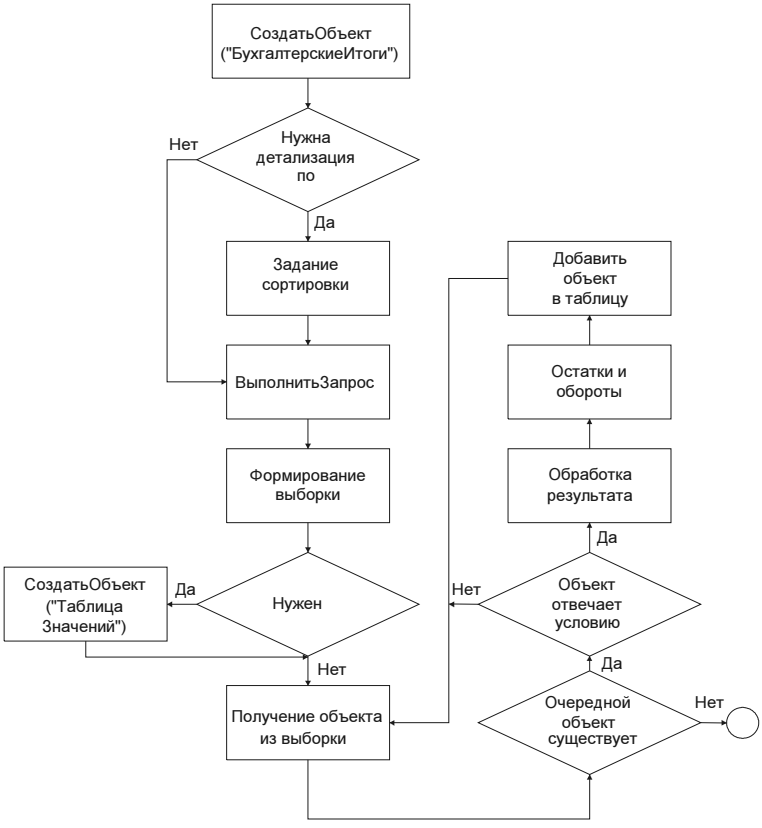


Рис. 3.25. Схема алгоритма получения бухгалтерских итогов в режиме запроса

Как видно из схемы (см. рис. 3.25), для получения бухгалтерских итогов в режиме запроса, если это необходимо, надо задать режим сортировки по субсчетам и субконто одним из методов, которые рассматриваются в *разд. "Задание сортировки"*. Отбор и расчет итогов по заданным параметрам выполняет метод `ВыполнитьЗапрос()`, который рассматривается в *разд. "ВыполнитьЗапрос"*. Выборка и получение объектов из выборки достигается методами, описанными в *разд. "Выборка результатов"*. В *разделах "Обращение к результатам"*, *"Остатки и обороты"*, *"Развернутое сальдо"* описаны методы обработки итогов.

Задание сортировки

1. `ВключатьСубсчета()`.

Метод `ВключатьСубсчета(ФлагСчета,ФлагКоррСчета)` позволяет установить режим отбора итогов методом `ВыполнитьЗапрос()` по субсчетам.

Параметры:

- `ФлагСчета` — признак развертывания сальдо по субсчетам основного счета (необязательный параметр), принимающий значение:
  - 0 — не разворачить по субсчетам;
  - 1 — разворачивать по субсчетам;
  - −1 (минус единица) — не выдавать итоги по счетам-группам (значение по умолчанию — 0);
- `ФлагКоррСчета` — признак развертывания сальдо по субсчетам корреспондирующего счета (необязательный параметр), принимающий значение:
  - 0 — не разворачить по субсчетам;
  - 1 — разворачивать по субсчетам;
  - −1 (минус единица) — не выдавать итоги по корреспондирующим счетам-группам (значение по умолчанию — 0).

2. `Опции()`.

Метод `Опции(ВклЗабалансСуммы,ВклОборотСубкСуммы)` позволяет установить режим включения сумм в итоги.

Параметры:

- `ВклЗабалансСуммы` — признак включения в итоги сумм по забалансовым счетам, принимающий значения:
  - 0 — не включаются суммы по забалансовым счетам;
  - 1 — включаются суммы по забалансовым счетам.
- `ВклОборотСубкСуммы` — признак включения в итоги сумм по оборотным субконто, принимающий значения:
  - 0 — не включаются суммы по оборотным субконто;
  - 1 — включаются суммы по оборотным субконто.

3. `ИспользоватьСубконто()`.

Метод `ИспользоватьСубконто(ВидСубконто,Субконто,ТипФильтра, ПоГруппам)` позволяет установить режим получения итогов методом `ВыполнитьЗапрос()` в отношении субконто. Метод следует вызывать до вызова метода `ВыполнитьЗапрос()`. Также он может вызываться последовательно несколько раз. В этом случае установки, выполняемые этим методом, суммируются.

Параметры:

- `ВидСубконто` — значение типа "ВидСубконто" (расчет временных итогов будет выполнен только для субконто указанного вида);
- `Субконто` — значение Субконто, по которому будут отобраны итоги по аналитике (если параметр не задан — то значение субконто считается пустым);
- `ТипФильтра` — число, определяющее тип фильтра по субконто и имеющее значение:
  - 1 — разворачивать по данному субконто;
  - 2 — отбирать по данному субконто;
  - 3 — не учитывать это субконто вообще (по умолчанию — 1);
- `ПоГруппам` — число, определяющее группировку итогов по субконто и принимающее значения:
  - 0 — не показывать итоги по группам справочника;
  - 1 — показывать итоги по группам справочника (по умолчанию — 0).

Внимание

Параметр `ПоГруппам` имеет смысл, если параметр `ТипФильтра` равен 1, а вид субконто, заданный параметром `ВидСубконто`, имеет тип значения "Справочник".

4. `ИспользоватьКорСубконто()`.

Метод `ИспользоватьКорСубконто(ВидСубконто,Субконто,ТипФильтра, ПоГруппам)` позволяет установить режим получения итогов методом ВыполнитьЗапрос() в отношении корреспондирующих субконто. Его следует вызывать до вызова метода `ВыполнитьЗапрос()`. Он также может вызываться последовательно несколько раз. В этом случае установки, выполняемые этим методом, суммируются.

Параметры:

- `ВидСубконто` — значение типа "ВидСубконто", определяющее расчет временных итогов только для субконто указанного вида;
- `Субконто` — значение Субконто, по которому будут отобраны итоги по аналитике (если параметр не задан, то значение субконто считается пустым);

- ТипФильтра — число, определяющее тип фильтра по субконто, принимающее значение:
  - 1 — разворачивать по данному субконто;
  - 2 — отбирать по данному субконто;
  - 3 — не учитывать это субконто вообще (по умолчанию — 1);
- ПоГруппам — число, определяющее группировку итогов по субконто и принимающее значение:
  - 0 — не показывать итоги по группам справочника;
  - 1 — показывать итоги по группам справочника (по умолчанию — 0).

Внимание

Параметр ПоГруппам имеет смысл, если параметр ТипФильтра равен 1, а вид субконто, заданный параметром ВидСубконто, имеет тип значения "Справочник".

5. ВыполнитьЗапрос().

Метод ВыполнитьЗапрос(НачалоПериода,КонецПериода,Счет,КоррСчет,Валюта, ТипИтогов,Периодичность,ТипСуммы) позволяет выполнить отбор и расчет итогов по заданным параметрам.

Возвращает значение:

- 1 — запрос выполнен;
- 0 — запрос не выполнен.

Параметры:

- НачалоПериода — дата, документ или позиция начала периода запроса;
- КонецПериода — дата, документ или позиция конца периода запроса;
- Счет — счета, для которых будут отбираться итоги в запросе (необязательный параметр, задается значением типа "Счет" или объектом типа "СписокЗначений", содержащим значения типа "Счет", либо строкой, содержащей список кодов счетов, разделенных символом запятая или точка с запятой), причем, если параметр не указан, отбор будет выполняться по всем счетам;
- КоррСчет — значение типа "Счет", представляющее корреспондирующий счет, в корреспонденции с которым будут отбираться итоги счета, указанного в параметре Счет (необязательный параметр, задается значением типа "Счет" или объектом типа "СписокЗначений", содержащим значения типа "Счет", либо строкой, содержащей список кодов счетов, разделенных символом запятая или точка с запятой), причем, если параметр не указан, будут отбираться итоги в корреспонденции по всем счетам;
- Валюта — зачение типа "Справочник.Валюты" (если параметр не указан, то итоги выдаются без учета валюты);

- `ТипИтогов` — число, определяющее тип отбираемых итогов и принимающее значение:
  - 1 — остатки и обороты по счету в целом;
  - 2 — обороты между счетами;
  - 3 — первое и второе вместе (по умолчанию — 1).
- `Периодичность` — число или символьная строка, позволяющие получить дополнительные данные по итогам на периоды (по умолчанию периодичность не задана);
- `ТипСуммы` — число или строка, определяющие тип рассчитываемых итогов (в скобках указаны строковые синонимы):
  - 1 (`С` или `S`) — рассчитывать суммы;
  - 2 (`В` или `С`) — рассчитывать валютные суммы;
  - 4 (`К` или `А`) — рассчитывать количество.

Внимание

Если требуется одновременно рассчитывать разные суммы, значение параметра получается путем сложения допустимых значений, например: 5 (1+4) — определяет рассчитывать суммы и количество.

При указании параметра строкой в ней указываются все символы, которые обозначают типы сумм (которые нужно рассчитывать). По умолчанию рассчитываются все типы сумм.

Введем в нашем документе "Авансовый отчет" (рис. 3.21) кнопку **Остаток** для расчета входящего остатка по сотруднику с помощью процедуры `РассчитатьПредыдущийОстаток()`. Текст этой процедуры приводится ниже.

```
Процедура РасчитатьПредыдущийОстаток()
  Если ДатаДок > КонецРассчитанногоПериодаБИ() Тогда
    Предупреждение("На " + ДатаДок + " бухгалтерские итоги не рассчитаны!
                  |Расчет итогов выполняется в режиме "Операции -
                  |Управление бухгалтерскими итогами"".");
  ИначеЕсли (Выбран() = 0) или ((Выбран() = 1) и
            (ДатаДок <> ТекущийДокумент().ДатаДок)) Тогда
    Предупреждение("Для получения остатка предыдущего аванса по данным
    |бухгалтерского учета документ необходимо записать.");
  Иначе
    БухИт = СоздатьОбъект("БухгалтерскиеИтоги");
// Зададим условие отбора по сотруднику Сотрудник
БухИт.ИспользоватьСубконто(ВидыСубконто.Сотрудники,Сотрудник,2);
```

```
    Если ТипОтчета = 1 Тогда
// Отберем все итоги за период от даты текущего документа до
// конца по счету 71.1
        БухИт.ВыполнитьЗапрос(ТекущийДокумент(),,"71.1");
// получим дебетовое и кредитовое сальдо на дату документа
        ОстатокПоСчету = БухИт.СНД("С") - БухИт.СНК("С");
    Иначе
// Отберем все итоги за период от даты текущего документа до конца
// по счету 71.11 и по реквизиту — Валюта
        БухИт.ВыполнитьЗапрос(ТекущийДокумент(),,"71.11",,Валюта);
        ОстатокПоСчету = БухИт.СНД("В") - БухИт.СНК("В");
    КонецЕсли;
    ОстатокПоСчету = ОстатокПоСчету - Сумма1 - Сумма2 - Сумма3;
    Если ОстатокПоСчету < 0 Тогда
        ТипОстатка = 2;
// Перерасход
        ПредОстаток = - ОстатокПоСчету;
    Иначе
        ТипОстатка = 1;
// Остаток
        ПредОстаток = ОстатокПоСчету;
    КонецЕсли;
  КонецЕсли;
КонецПроцедуры    //РасчитатьПредыдущийОстаток
```

Выборка результатов

1. `ВыбратьСчета()`.

Метод `ВыбратьСчета(ФлагВсе,ФлагДК,Номер,РазвСальдо)` открывает выборку счетов, для которых были получены итоги методом "ВыполнитьЗапрос".

Возвращает значение:

- 1 — если действие выполнено и в выборке есть хотя бы один счет;
- 0 — если действие не выполнено или в выборке нет ни одного счета.

Параметры:

- `ФлагВсе`:
  - 0 — отбирать те счета, которые имели итоги на этом уровне обхода итогов запроса;
  - 1 — включить в выборку все счета, которые имели итоги в данном запросе;

- −1, −2 — включить в выборку счета, которые имели итоги в группировке вышестоящего уровня n (по умолчанию — 0);

❒ `ФлагДК`:

- 1 — включать в выборку счета только с дебетовыми оборотами;
- 2 — включать в выборку счета только с кредитовыми оборотами;
- 0 — включать в выборку счета вне зависимости от дебетовых или кредитовых оборотов (по умолчанию — 0);

❒ `Номер` — число, определяющее номер выборки (если параметр не указан, выборке присваивается номер 0);

❒ `РазвСальдо` — число, опрделяющее признак необходимости рассчитывать развернутое сальдо по субконто (используется только в том случае, если в запросе участвуют субконто) и принимающее значение:

- 1 — рассчитывать развернутое сальдо;
- 0 — не рассчитывать развернутое сальдо (по умолчанию 0).

2. `ПолучитьСчет()`.

Метод `ПолучитьСчет(Номер,Счет)` позволяет получить из выборки следующий счет. Выборка должна быть предварительно открыта при помощи метода "ВыбратьСчета".

Возвращает значение:

❒ 1 — следующий счет выбран успешно;

❒ 0 — следующий счет не выбран (отсутствует).

Параметры:

❒ `Номер` — число, определяющее номер выборки (необязательный параметр);

❒ `Счет` — значение счета, на которое нужно спозиционироваться.

3. `ВыбратьКорСчета()`.

Метод `ВыбратьКорСчета(ФлагВсе,ФлагДК,Номер)` открывает выборку корреспондирующих счетов, для которых были получены итоги методом "ВыполнитьЗапрос".

Возвращает значение:

❒ 1 — если действие выполнено и в выборке есть хотя бы один счет;

❒ 0 — если действие не выполнено или в выборке нет ни одного счета.

Параметры:

❒ `ФлагВсе`:

- 0 — отбирать те счета, которые имели итоги на этом уровне обхода итогов запроса;

- 1 — включить в выборку все счета, которые имели итоги в данном запросе;
- −1, −2 — включить в выборку счета, которые имели итоги в группировке вышестоящего уровня n (по умолчанию — 0);

❐ ФлагДК:

- 1 — включать в выборку счета только с дебетовыми оборотами;
- 2 — включать в выборку счета только с кредитовыми оборотами;
- 0 — включать в выборку счета вне зависимости от дебетовых или кредитовых оборотов (по умолчанию — 0);

❐ Номер — число, определяющее номер выборки (если параметр не указан, выборке присваивается номер 0).

4. ПолучитьКорСчет().

Метод ПолучитьКорСчет(Номер,Счет) позволяет получить из выборки следующий корреспондирующий счет. Выборка должна быть предварительно открыта при помощи метода ВыбратьКорСчета().

Возвращает значения:

❐ 1 — следующий счет выбран успешно;

❐ 0 — следующий счет не выбран (отсутствует).

Параметры:

❐ Номер — число, представляющее номер выборки (необязательный параметр);

❐ Счет — значение счета, на которое нужно спозиционироваться.

5. ВыбратьВалюты().

Метод ВыбратьВалюты(ФлагВсе,ФлагДК,Номер,РазвСальдо,Сортировка) открывает выборку валют.

Возвращает значение:

❐ 1 — если действие выполнено и в выборке есть хотя бы одна валюта;

❐ 0 — если действие не выполнено или в выборке нет ни одной валюты.

Параметры:

❐ ФлагВсе:

- 0 — отбирать те счета, которые имели итоги на этом уровне обхода итогов запроса;
- 1 — включить в выборку все счета, которые имели итоги в данном запросе;
- −1, −2 — включить в выборку счета, которые имели итоги в группировке вышестоящего уровня n (по умолчанию — 0);

- ❒ `ФлагДК`:
  - 1 — включать в выборку счета только с дебетовыми оборотами;
  - 2 — включать в выборку счета только с кредитовыми оборотами;
  - 0 — включать в выборку счета вне зависимости от дебетовых или кредитовых оборотов (по умолчанию 0);
- ❒ `Номер` — число, определяющее номер выборки (если параметр не указан, выборке присваивается номер 0);
- ❒ `РазвСальдо` — признак необходимости рассчитывать развернутое сальдо по субконто (используется, только если в запросе участвуют субконто):
  - 1 — рассчитывать развернутое сальдо;
  - 0 — не рассчитыть развернутое сальдо (по умолчанию 0);
- ❒ `Сортировка` — строка, представляющая идентификатор реквизита справочника валют, который будет использован для упорядочивания обхода валют методом `ПолучитьВалюту()` (если значение пустое, используется представление справочника).

6. `ПолучитьВалюту()`.

Метод `ПолучитьВалюту(Номер,Валюта)` позволяет получить из выборки следующую валюту. Выборка должна быть предварительно открыта при помощи метода `ВыбратьВалюты()`.

Возвращает значения:

- ❒ 1 — следующая валюта выбрана успешно;
- ❒ 0 — следующая валюта не выбрана (отсутствует).

Параметры:

- ❒ `Номер` — число, представляющее номер выборки (необязательный параметр);
- ❒ `Валюта` — значение валюты, на которое нужно спозиционироваться.

7. `ВыбратьПериоды()`.

Метод `ВыбратьПериоды(ФлагВсе,ФлагДК,Номер,РазвСальдо)` открывает выборку периодов.

Возвращает значения:

- ❒ 1 — если действие выполнено и в выборке есть хотя бы один период;
- ❒ 0 — если действие не выполнено или в выборке нет ни одного периода.

Параметры:

- ❒ `ФлагВсе`:
  - 0 — отбирать те счета, которые имели итоги на этом уровне обхода итогов запроса;

- 1 — включить в выборку все счета, которые имели итоги в данном запросе;
- −1, −2 — включить в выборку счета, которые имели итоги в группировке вышестоящего уровня n (по умолчанию — 0);

❒ ФлагДК:

- 1 — включать в выборку счета только с дебетовыми оборотами;
- 2 — включать в выборку счета только с кредитовыми оборотами;
- 0 — включать в выборку счета вне зависимости от дебетовых или кредитовых оборотов (по умолчанию — 0);

❒ Номер — число, определяющее номер выборки (если параметр не указан, выборке присваивается номер 0);

❒ РазвСальдо — признак необходимости рассчитывать развернутое сальдо по субконто (используется, только если в запросе участвуют субконто):

- 1 — рассчитывать развернутое сальдо;
- 0 — не рассчитыть развернутое сальдо (по умолчанию — 0).

8. ПолучитьПериод().

Метод ПолучитьПериод(Номер,ДатаНачалаПериода) позволяет получить из выборки следующий период. Выборка должна быть предварительно открыта при помощи метода ВыбратьПериоды().

Возвращает значения:

❒ 1 — следующий период выбран успешно;

❒ 0 — следующий период не выбран (отсутствует).

Параметры:

❒ Номер — число, определяющее номер выборки (необязательный параметр);

❒ ДатаНачалаПериода — значение даты начала периода, на который нужно спозиционироваться.

9. ВыбратьСубконто().

Метод ВыбратьСубконто(Индекс,ФлагВсе,ФлагДК,Номер,РазвСальдо,Сортировка, ОбратныйПорядок) открывает выборку по субконто.

Возвращает:

❒ 1 — если действие выполнено и в выборке есть хотя бы одно субконто;

❒ 0 — если действие не выполнено или в выборке нет ни одного субконто.

Параметры:

❒ Индекс — число, представляющее порядковый номер вызова метода ИспользоватьСубконто();

- ФлагВсе:
  - 0 — отбирать те субконто, которые имели итоги на этом уровне обхода итогов запроса;
  - 1 — включить в выборку все субконто, которые имели итоги в данном запросе;
  - −1, −2 — включить в выборку субконто, которые имели итоги в группировке вышестоящего уровня n (по умолчанию — 0);
- ФлагДК:
  - 1 — включать в выборку счета только с дебетовыми оборотами;
  - 2 — включать в выборку счета только с кредитовыми оборотами;
  - 0 — включать в выборку счета вне зависимости от дебетовых или кредитовых оборотов (по умолчанию — 0);
- Номер — число, представляющее номер выборки (если параметр не указан, выборке присваивается номер 0);
- РазвСальдо — признак необходимости рассчитывать развернутое сальдо по субконто (используется только в случае участия в запросе субконто):
  - 1 — рассчитывать развернутое сальдо;
  - 0 — не рассчитыть развернутое сальдо (по умолчанию — 0);
- Сортировка — строка, определяющая идентификатор реквизита субконто (если субконто — справочник или документ или счет), который будет использован для упорядочивания обхода субконто методом ПолучитьСубконто(), при этом если значение "пустое", то используется стандартное представление;
- ОбратныйПорядок:
  - 0 — выборка в прямом порядке;
  - 1 — выборка в обратном порядке (по умолчанию — 0).

10. ПолучитьСубконто().

Метод ПолучитьСубконто(Индекс,Номер,Значение) позволяет получить из выборки следующее субконто. Выборка должна быть предварительно открыта при помощи метода ВыбратьСубконто().

Возвращает значения:

- 1 — следующее субконто выбрано успешно;
- 0 — следующее субконто не выбрано (отсутствует).

Параметры:

- Индекс — число, определяющее порядковый номер вызова метода ИспользоватьСубконто();

- Номер — число, представляющее номер выборки (необязательный параметр);
- Значение — значение субконто, на которое нужно спозиционироваться.

Внимание

Возможно использование данного метода совместно с параметром ВидСубконто вместо Индекс. Параметр Номер при этом можно не задавать, например, БухИт.ПолучитьСубконто(ВидыСубконто.Оборудование, ТаблицаОстатков.Оборудование).

11. ВыбратьКорСубконто().

Метод ВыбратьКорСубконто(Индекс,ФлагВсе,ФлагДК,Номер,Сортировка, ОбратныйПорядок) открывает выборку по корреспондирующим субконто.

Возвращает значение:

- 1 — если действие выполнено и в выборке есть хотя бы одно субконто;
- 0 — если действие не выполнено или в выборке нет ни одного субконто.

Параметры:

- Индекс — число, представляющее порядковый номер вызова метода ИспользоватьКорСубконто();
- ФлагВсе:
  - 0 — отбирать те субконто, которые имели итоги на этом уровне обхода итогов запроса;
  - 1 — включить в выборку все субконто, которые имели итоги в данном запросе;
  - −1, −2 — включить в выборку субконто, которые имели итоги в группировке вышестоящего уровня n (по умолчанию — 0);
- ФлагДК:
  - 1 — включать в выборку счета только с дебетовыми оборотами;
  - 2 — включать в выборку счета только с кредитовыми оборотами;
  - 0 — включать в выборку счета вне зависимости от дебетовых или кредитовых оборотов (по умолчанию — 0);
- Номер — число, определяющее номер выборки (если параметр не указан, выборке присваивается номер 0);
- Сортировка — строка, представляющая идентификатор реквизита субконто (если субконто — справочник, документ или счет), который будет использован для упорядочивания обхода субконто методом ПолучитьКорСубконто(), причем если значение пустое, то используется стандартное представление;

- ОбратныйПорядок:
  - 1 — выборка в прямом порядке;
  - −1 — выборка в обратном порядке; необязательный параметр (по умолчанию — 1).

12. ПолучитьКорСубконто().

Метод ПолучитьКорСубконто(Индекс,Номер,Значение) позволяет получить из выборки следующее субконто. Выборка должна быть предварительно открыта при помощи метода ВыбратьКорСубконто.

Возвращает значение:

- 1 — следующее субконто выбрано успешно;
- 0 — следующее субконто не выбрано (отсутствует).

Параметры:

- Индекс — число, определяющее порядковый номер вызова метода ИспользоватьКорСубконто();
- Номер — число, определяющее номер выборки (необязательный параметр);
- Значение — значение корсубконто, на которое нужно спозиционироваться.

Обращение к результатам

Для обращения к результатам используются следующие методы:

1. Субконто().

Метод Субконто(Номер|ВидСубконто) возвращает значение субконто, соответствующее текущему итогу.

Параметр — Номер или ВидСубконто определяет соответственно номер выборки субконто или значение типа "Вид субконто".

2. КорСубконто().

Метод КорСубконто(Номер|ВидСубконто) возвращает значение корреспондирующего субконто, соответствующее текущему итогу.

Параметр — Номер или ВидСубконто определяет соответственно номер выборки корреспондирующего субконто или значение типа "ВидСубконто".

3. ПредставлениеСубконто().

Метод ПредставлениеСубконто(Номер|ВидСубконто,Краткое) возвращает строковое представление для субконто, соответствующего текущему итогу.

Параметры:

- Номер (ВидСубконто) — определяет номер выборки субконто (значение типа "ВидСубконто");

- Краткое — число, принимающее значения:
  - 0 — полное представление субконто;
  - 1 — краткое представление субконто (по умолчанию — 0).

Краткое представление субконто означает представление значения субконто соответствующего типа, заданное в Конфигураторе, либо формируемое в соответствии с правилами преобразования типов. Например:

- для субконто типа "Справочник" краткое представление будет зависеть от свойства "Основное представление", определенного для этого справочника;
- для субконто типа "Перечисление" краткое представление (впрочем, как и полное) будет представлением значения перечисления;
- для субконто типа "Документ" кратким представлением будет идентификатор или синоним документа, его номер и дата.

Полное представление субконто — это представление, заданное в свойствах данного вида субконто в Конфигураторе. Для задания представления используются управляющие элементы закладки "Представление" в палитре свойств.

4. ПредставлениеКорСубконто().

Метод ПредставлениеКорСубконто(Номер|ВидСубконто,Краткое) возвращает строковое представление для корреспондирующего субконто, соответствующего текущему итогу.

Параметры:

- Номер (ВидСубконто) — номер выборки корреспондирующего субконто (значение типа "ВидСубконто");
- Краткое:
  - 0 — полное представление субконто;
  - 1 — краткое представление субконто (по умолчанию — 0).

Краткое представление субконто означает представление значения субконто соответствующего типа, заданное в Конфигураторе, либо формируемое в соответствии с правилами преобразования типов. Например, для субконто типа "Справочник" краткое представление будет зависеть от свойства "Основное представление", определенного для этого справочника; для субконто типа "Перечисление" краткое представление (впрочем, как и полное), будет представлением значения перечисления; для субконто типа "Документ" кратким представлением будет идентификатор или синоним документа, его номер и дата. Полное представление субконто — это представление, заданное в свойствах данного вида субконто в Конфигураторе. Для задания представления используются управляющие элементы закладки **Представление** в палитре свойств.

5. `ЭтоГруппа()`.

Метод `ЭтоГруппа()` определяет, является ли текущее значение группировки запроса по субконто группой или нет. Имеет смысл только для субконто типа "Справочник".

Возвращает:

- 1 — значение группировки по субконто является группой;
- 0 — значение группировки по субконто не является группой.

6. `ВыбранаПоДт()` .

Метод `ВыбранаПоДт()` определяет, выбран ли итог по дебету.

Возвращает:

- 0 — данный итог по дебету не выбран;
- 1 — данный итог по дебету выбран.

7. `ВыбранаПоКт()`.

Метод `ВыбранаПоДт()` определяет, выбран ли итог по кредиту.

Возвращает:

- 0 — данный итог по кредиту не выбран;
- 1 — данный итог по кредиту выбран.

Остатки и обороты

1. `СНД()`.

Метод `СНД(ТипСуммы)` возвращает дебетовое сальдо на начало периода.

Параметр:

- `ТипСуммы` — число или строка (необязательный параметр), определяющие тип возвращаемой суммы и принимающие значение:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;
  - 3 (`К`) — количество (если параметр не указан, метод возвращает сумму).

2. `СНК()`.

Метод `СНК(ТипСуммы)` возвращает кредитовое сальдо на начало периода.

Параметры:

- `ТипСуммы` — число или строка (необязательный параметр), определяющие тип возвращаемой суммы и принимающие одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;

- 3 (К) — количество (если параметр не указан, метод возвращает сумму).

3. СКД().

Метод СКД(ТипСуммы) возвращает дебетовое сальдо на конец периода.

Параметры:

- ТипСуммы — число или строка (необязательный параметр), определяющие тип возвращаемой суммы и принимающие одно из следующих значений:
  - 1 (С) — сумма;
  - 2 (В) — валютная сумма;
  - 3 (К) — количество (если параметр не указан, метод возвращает сумму).

4. СКК().

Метод СКК(ТипСуммы) возвращает кредитовое сальдо на конец периода.

Параметры:

- ТипСуммы — число или строка (необязательный параметр), определяющие тип возвращаемой суммы, принимающие одно из следующих значений:
  - 1 (С) — сумма;
  - 2 (В) — валютная сумма;
  - 3 (К) — количество (если параметр не указан, метод возвращает сумму).

5. ДО().

Метод ДО(ТипСуммы) возвращает дебетовый оборот на начало периода.

Параметры:

- ТипСуммы — число или строка (необязательный параметр), определяющие тип возвращаемой суммы, принимающие одно из следующих значений:
  - 1 (С) — сумма;
  - 2 (В) — валютная сумма;
  - 3 (К) — количество (если параметр не указан, метод возвращает сумму).

6. КО().

Метод КО(ТипСуммы) возвращает кредитовый оборот на начало периода.

Параметры:

- ТипСуммы — число или строка (необязательный параметр), определяющие тип возвращаемой суммы, принимающие одно из следующих значений:
  - 1 (С) — сумма;
  - 2 (В) — валютная сумма;
  - 3 (К) — количество (если параметр не указан, метод возвращает сумму).

Развернутое сальдо

1. `СНДРС()`.

Метод `СНДРС(ТипСуммы)` возвращает дебетовое развернутое сальдо на начало периода.

Параметры:

- `ТипСуммы` — число или строка (необязательный параметр), определяющие тип возвращаемой суммы, принимающие одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;
  - 3 (`К`) — количество (если параметр не указан, метод возвращает сумму).

2. `СНКРС()`.

Метод `СНКРС(ТипСуммы)` возвращает кредитовое развернутое сальдо на начало периода.

Параметры:

- `ТипСуммы` — число или строка (необязательный параметр), определяющие тип возвращаемой суммы, принимающие одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;
  - 3 (`К`) — количество (если параметр не указан, метод возвращает сумму).

3. `СКДРС()`.

Метод `СКДРС(ТипСуммы)` возвращает дебетовое развернутое сальдо на конец периода.

Параметры:

- `ТипСуммы` — число или строка (необязательный параметр), определяющие тип возвращаемой суммы, принимающие одно из следующих значений:
  - 1 (`С`) — сумма;
  - 2 (`В`) — валютная сумма;
  - 3 (`К`) — количество (если параметр не указан, метод возвращает сумму).

4. `СККРС()`.

Метод `СККРС(ТипСуммы)` возвращает кредитовое развернутое сальдо на конец периода.

Параметры:

- `ТипСуммы` — число или строка (необязательный параметр), определяющие тип возвращаемой суммы, принимающие одно из следующих значений:
  - 1 (`С`) — сумма;

- 2 (в) — валютная сумма;
- 3 (к) — количество (если параметр не указан, метод возвращает сумму).

Создадим в нашей конфигурации документ "Передача оборудования в монтаж" (рис. 3.26).

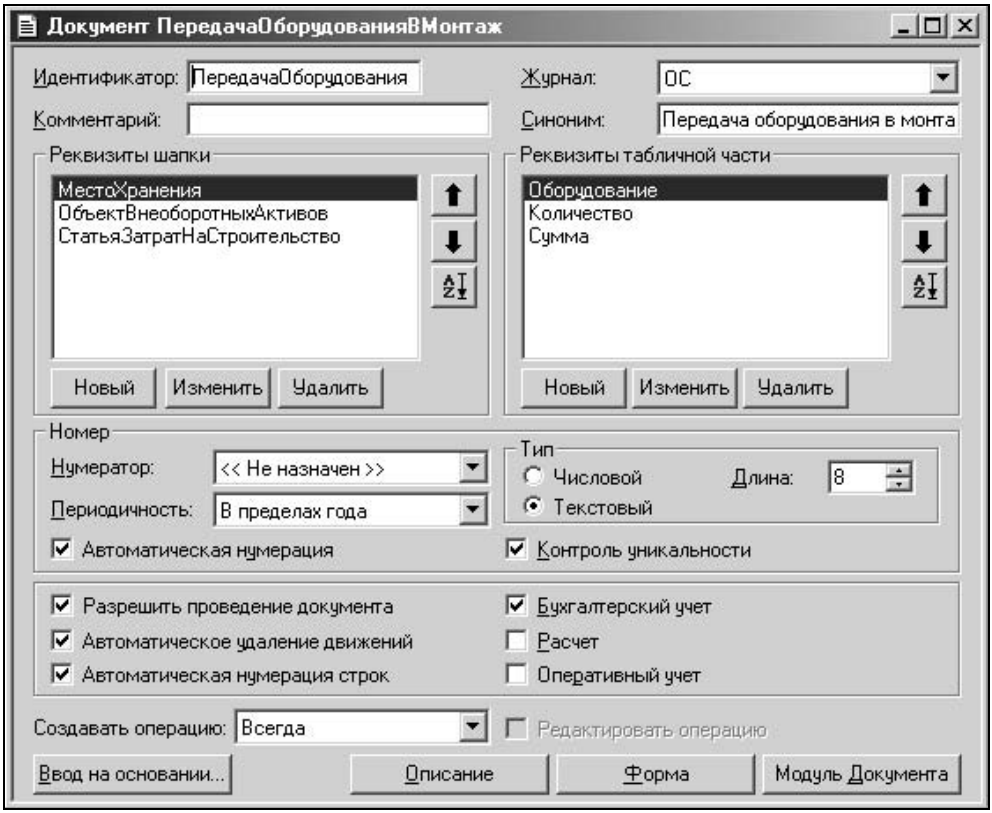


Рис. 3.26. Диалоговое окно документа "Передача оборудования в монтаж"

В предопределенной процедуре `ОбработкаПроведения()` нашего документа заполним таблицу остатков оборудования на складе для дальнейшего формирования корректных проводок:

```
// Создадим объект типа «БухгалтерскиеИтоги»
БухИт = СоздатьОбъект("БухгалтерскиеИтоги");
// Зададим отбор по субконто первого уровня вида "Оборудование" со
// значением из списка – СписокОборудования
БухИт.ИспользоватьСубконто(ВидыСубконто.Оборудование, СписокОборудования,2);
// по всем складам
```

```
БухИт.ИспользоватьСубконто();
// Отберем по запросу суммовые и количественные итоги за весь
//период до текущего
// документа по счетам 07 "Оборудование к установке",
// Н01.02 "Учет стоимости оборудования", НПР07 "Оборудование"
БухИт.ВыполнитьЗапрос(,ТекущийДокумент(),
                              "07, Н01.02, НПР.07",,,,,"СК");
// Создадим объект типа «ТаблицаЗначений»
ТаблицаОстатков = СоздатьОбъект("ТаблицаЗначений");
// Выгрузим из табличной части документа в нашу таблицу
// колонки Оборудование и Количество
ВыгрузитьТабличнуюЧасть(ТаблицаОстатков,
                                   "Оборудование, Количество");
ТаблицаОстатков.Свернуть("Оборудование", "Количество");
// Добавим колонки в нашу таблице для итогов
ТаблицаОстатков.НоваяКолонка("КоличествоНаСкладах", "Число");
ТаблицаОстатков.НоваяКолонка("СуммаНаСкладах", "Число");
ТаблицаОстатков.НоваяКолонка("СредняяСтоимость", "Число");
ТаблицаОстатков.НоваяКолонка("СуммаПР", "Число");
ТаблицаОстатков.НоваяКолонка("КоличествоНаСкладахН", "Число");
ТаблицаОстатков.НоваяКолонка("СуммаНаСкладахН", "Число");
ТаблицаОстатков.ВыбратьСтроки();
// Заполним нашу таблицуитогами
Пока ТаблицаОстатков.ПолучитьСтроку() = 1 Цикл
    Если ТаблицаОстатков.Количество = 0 Тогда
       ТекстСообщения = "Не указано количество оборудования: "+
                    ТаблицаОстатков.Оборудование;
       Возврат;
    КонецЕсли;
    КоличествоНаСкладе = 0;
// Получим из выборки объект на первом уровне - текущее оборудование
    Если БухИт.ПолучитьСубконто(ВидыСубконто.Оборудование,, Табли-
        цаОстатков.Оборудование) = 1 Тогда
// Переберем все движения данного оборудования по счетам
       БухИт.ВыбратьСчета();
       Пока БухИт.ПолучитьСчет() = 1 Цикл
          Если БухИт.Счет.Код = "07" Тогда
// Суммовой остаток, потребуется для расчета средней стоимости
             ТаблицаОстатков.СуммаНаСкладах = БухИт.СКД("С");
```

```
            ТаблицаОстатков.КоличествоНаСкладах = БухИт.СКД("К");
// Получим из выборки объект на втором уровне - на заданном  складе
Если БухИт.ПолучитьСубконто(2,, МестоХранения) = 1 Тогда
                КоличествоНаСкладе = БухИт.СКД("К");
            КонецЕсли;
         ИначеЕсли БухИт.Счет.Код = "НПР.07" Тогда
            ТаблицаОстатков.СуммаПР = БухИт.СКД("С");
         Иначе // счет = Н01.02
            ТаблицаОстатков.СуммаНаСкладахН = БухИт.СКД("С");
            ТаблицаОстатков.КоличествоНаСкладахН = БухИт.СКД("К");
         КонецЕсли;
      КонецЦикла;
   КонецЕсли;
   Если КоличествоНаСкладе < ТаблицаОстатков.Количество Тогда
      ТекстСообщения = "На складе "+КоличествоНаСкладе+" единиц "+"
      из необходимых "+ТаблицаОстатков.Количество+" единиц оборудо-
      вания "+ТаблицаОстатков.Оборудование;
      Возврат;
   КонецЕсли;
ТаблицаОстатков.СредняяСтоимость = ТаблицаОстат-
ков.СуммаНаСкладах/ТаблицаОстатков.КоличествоНаСкладах;
Стр = ""+ТаблицаОстатков.НомерСтроки+"."+
ТаблицаОстатков.Оборудование+" ";
Стр = Стр+": на складе "+МестоХранения+" "+КоличествоНаСкладе+" единиц,";
Стр = Стр+" всего на складах "+ТаблицаОстатков.КоличествоНаСкладах+"
единиц на сумму "+ТаблицаОстатков.СуммаНаСкладах+",";
Стр = Стр+" средняя стоимость "+
Окр(ТаблицаОстатков.СредняяСтоимость, 2, 1);
ТекстСообщения = Стр;
глСообщениеПроведения(ТекстСообщения, ТекущийДокумент(), 0);
КонецЦикла;
```

Методы таблиц значений

В предыдущих разделах неоднократно упоминалось об использовании таблицы значений. Рассмотрим схему применения методов объектов типа "ТаблицаЗначений" (рис. 3.27).

Для задания структуры таблицы используются методы, изложенные в разделе "Методы задания структуры таблицы". Кроме этих методов структура таблицы задается методом `Загрузить(ТаблицаЗначений)` путем загрузки структуры из загружаемой таблицы. Для формирования строк таблицы используются

методы, изложенные в *разд. "Методы формирования строк таблицы"*. Кроме этих методов строки таблицы формируются методом `Загрузить()` и `Заполнить()` путем копирования строк из другой таблицы значений или из списка значений. Особый интерес представляют методы `Итог()`, `Сортировать()` и `Свернуть()`. Данные методы рассматриваются в *разд. "Методы работы со всей таблицей"*.

Рис. 3.27. Схема работы с таблицей значений

Методы задания структуры таблицы

1. `КоличествоКолонок()`.

Метод `КоличествоКолонок(Колич)` устанавливает или возвращает количество колонок.

Параметр:

`Колич` — числовое выражение (необязательный параметр), которое указывает новое количество колонок.

2. `НоваяКолонка()`.

Метод `НоваяКолонка(Идентификатор,Тип,Длина,Точность,Заголовок,Ширина, Формат,Положение)` позволяет добавить в конец таблицы значений новую колонку.

Возвращает номер новой колонки.

Параметры:

- Идентификатор — идентификатор колонки (необязательный параметр), причем если данный параметр не указан, то обращение к колонке возможно только по номеру;
- Тип — строка или вид субконто, задающий тип колонки (необязательный параметр), причем если данный параметр не указан, то можно хранить любой тип;
- Длина — длина для числовой или строковой колонки (необязательный параметр);
- Точность — точность или длина дробной части для числовой колонки (необязательный параметр);
- Заголовок — строковое выражение, содержащее заголовок колонки в элементе диалога типа "ТаблицаЗначений" (необязательный параметр);
- Ширина — числовое выражение, содержащее ширину колонки (в символах) для представления колонки в элементе диалога типа "ТаблицаЗначений" (необязательный параметр);
- Формат — строковое выражение, содержащее форматированную строку, которая будет использована при визуальном отображении значений данной колонки (необязательный параметр);
- Положение — определяет вариант выравнивания при визуальном отображении значений данной колонки (необязательный параметр):
  - 1 — слева;
  - 2 — справа.

3. ВставитьКолонку().

Метод ВставитьКолонку(Идентификатор,НомерКолонки,Тип,Длина,Точность, Заголовок, Ширина,Формат,Положение) позволяет в указанную позицию таблицы значений вставить новую колонку.

Возвращает номер новой колонки.

Параметры:

- Идентификатор — идентификатор колонки (необязательный параметр), причем если данный параметр не указан, то обращение к колонке возможно только по номеру;
- НомерКолонки — числовое выражение, содержащее позицию, в которую вставляется новая колонка (необязательный параметр);
- Тип — строка или вид субконто, задающий тип колонки (необязательный параметр), причем если данный параметр не указан, то можно хранить любой тип;
- Длина — длина для числовой или строковой колонки (необязательный параметр);

- ❒ `Точность` — точность или длина дробной части для числовой колонки (необязательный параметр);
- ❒ `Заголовок` — строковое выражение, содержащее заголовок колонки в элементе диалога типа "ТаблицаЗначений" (необязательный параметр);
- ❒ `Ширина` — числовое выражение, содержащее ширину колонки (в символах) для представления колонки в элементе диалога типа "ТаблицаЗначений" (необязательный параметр);
- ❒ `Формат` — строковое выражение, содержащее форматированную строку, которая будет использована при визуальном отображении значений данной колонки (необязательный параметр);
- ❒ `Положение` — определяет вариант выравнивания при визуальном отображении значений данной колонки (необязательный параметр):
  - 1 — слева;
  - 2 — справа.

4. `УдалитьКолонку()`.

Метод `УдалитьКолонку(Колонка)` удаляет колонку из таблицы значений.

Параметр — `Колонка` определяет номер или идентификатор удаляемой колонки.

5. `УстановитьПараметрыКолонки()`.

Метод `УстановитьПараметрыКолонки(Колонка,Тип,Длина,Точность,Заголовок,Ширина,Формат,Положение)` позволяет установить новые значения параметров колонки (только те, которые указаны).

Если какой-либо параметр при вызове метода не задан, то данный параметр колонки не меняется.

Параметры:

- ❒ `Колонка` — номер или код колонки, для которой будут установлены новые параметры;
- ❒ `Тип` — строка или вид субконто, задающий тип колонки (необязательный параметр), причем если данный параметр не указан, то можно хранить любой тип;
- ❒ `Длина` — длина для числовой или строковой колонки (необязательный параметр);
- ❒ `Точность` — точность или длина дробной части для числовой колонки (необязательный параметр);
- ❒ `Заголовок` — строковое выражение, содержащее заголовок колонки в элементе диалога типа "ТаблицаЗначений" (необязательный параметр);

- Ширина — числовое выражение, содержащее ширину колонки (в символах) для представления ее в элементе диалога типа "ТаблицаЗначений" (необязательный параметр);
- Формат — строковое выражение, содержащее форматированную строку, которая будет использована при визуальном отображении значений данной колонки (необязательный параметр);
- Положение — определяет вариант выравнивания при визуальном отображении значений данной колонки (необязательный параметр):
  - 1 — слева;
  - 2 — справа.

6. ПолучитьПараметрыКолонки().

Метод ПолучитьПараметрыКолонки(Колонка,Тип,Длина,Точность,Заголовок,Ширина,Формат,Положение) позволяет получить значения параметров колонки. Возвращает номер или код колонки. Если в параметре Колонка задан номер колонки, то возвращается код колонки, и наоборот.

Параметры:

- Колонка — номер или код колонки, для которой требуется получить параметры;
- Тип — идентификатор переменной, в которую метод вернет строку, описывающую тип колонки или вид субконто (необязательный параметр);
- Длина — идентификатор переменной, в которую метод вернет длину для строковых и числовых значений (необязательный параметр);
- Точность — идентификатор переменной, в которую метод вернет точность для числовых значений колонки (необязательный параметр);
- Заголовок — идентификатор переменной, в которую метод вернет строку, описывающую заголовок колонки для показа (необязательный параметр);
- Ширина — идентификатор переменной, в которую метод вернет ширину колонки в таблице (необязательный параметр);
- Формат — идентификатор переменной, в которую метод вернет форматированную строку, которая используется при визуальном отображении значений данной колонки (необязательный параметр);
- Положение — идентификатор переменной, в которую метод вернет вариант выравнивания при визуальном отображении значений данной колонки (необязательный параметр):
  - 1 — слева;
  - 2 — справа.

В приведенном фрагменте для формирования бухгалтерских итогов использовался комбинированный способ задания структуры таблицы значений — ТаблицаОстатков:

```
// Создадим объект типа "ТаблицаЗначений"
ТаблицаОстатков = СоздатьОбъект("ТаблицаЗначений");
// 1 способ: выгрузка из табличной части документа в
// таблицу колонок
// Оборудование и Количество
ВыгрузитьТабличнуюЧасть(ТаблицаОстатков, "Оборудование, Количество");
// 2-й способ: Добавление колонок в таблицу
ТаблицаОстатков.НоваяКолонка("КоличествоНаСкладах", "Число");
ТаблицаОстатков.НоваяКолонка("СуммаНаСкладах", "Число");
ТаблицаОстатков.НоваяКолонка("СредняяСтоимость", "Число");
ТаблицаОстатков.НоваяКолонка("СуммаПР", "Число");
ТаблицаОстатков.НоваяКолонка("КоличествоНаСкладахН", "Число");
ТаблицаОстатков.НоваяКолонка("СуммаНаСкладахН", "Число");
```

Методы формирования строк таблицы

1. `НоваяСтрока()`.

Метод `НоваяСтрока(НомерСтроки)` добавляет новую строку в таблицу значений.

Параметр — `НомерСтроки`, представляет числовое выражение, содержащее позицию, в которую следует вставить новую строку (необязательный параметр).

2. `УдалитьСтроку()`.

Метод `УдалитьСтроку(НомерСтроки)` позволяет удалить строку из таблицы значений.

Параметр — `НомерСтроки`, определяет номер строки, которая будет удалена (если параметр не указан, то удаляется текущая строка).

3. `УдалитьСтроки()`.

Метод `УдалитьСтроки()` удаляет все строки из таблицы значений.

4. `ВыбратьСтроки()`.

Метод `ВыбратьСтроки()` открывает выборку строк таблицы значений. Дальнейшая выборка осуществляется при помощи метода `ПолучитьСтроку()`.

5. `ПолучитьСтроку()`.

Метод `ПолучитьСтроку()` позволяет получить из выборки следующую строку таблицы значений в последовательности выборки, открытой перед этим при помощи метода `ВыбратьСтроки()`.

Возвращает значения:

- 1 — если очередная строка выбрана;
- 0 — если не выбрана.

6. `ВыбратьСтроку()`.

Метод `ВыбратьСтроку(Строка,Заголовок,Таймаут)` позволяет открыть окно для интерактивного выбора строки в таблице значений.

Возвращает число:

- 1 — если выбор произведен (нажата кнопка OK);
- 0 — если выбор не произведен (нажата кнопка "ОТМЕНА");
- −1 (минус единица) — закончилось время (Таймаут) ожидания отклика пользователя.

Параметры:

- `Строка` — идентификатор переменной, куда помещается значение, определяющее номер выбранной строки (необязательный параметр, при вызове метода здесь можно передавать значение начального номера строки);
- `Заголовок` — строковое выражение, значение которого отображается в заголовке диалогового окна (необязательный параметр, может использоваться для подсказки пользователю);
- `Таймаут` — числовое выражение, значение которого задает время ожидания системы (в секундах) на отклик пользователя (необязательный параметр, если не задано, то время ожидания бесконечно).

7. `ПолучитьСтрокуПоНомеру()`.

Метод `ПолучитьСтрокуПоНомеру(НомерСтроки)` позволяет получить строку таблицы значений по номеру. Указанная строка становится текущей.

Параметр — `НомерСтроки` представляет номер строки, на которую следует переместиться.

8. `СдвинутьСтроку()`.

Метод `СдвинутьСтроку(КоличСтрок,НомерСтроки)` позволяет переместить строку таблицы значений на новую позицию.

Параметры:

- `КоличСтрок` — число строк, на которое надо переместить строку (если число положительное, то строка сдвигается вниз, если отрицательное, то вверх);
- `НомерСтроки` — номер строки, которую надо переместить (необязательный параметр, если параметр не задан, то принимается номер текущей строки).

Характерный пример сканирования таблицы значений:

```
ТаблицаОстатков.ВыбратьСтроки();
Пока ТаблицаОстатков.ПолучитьСтроку() = 1 Цикл
…
КонецЦикла;
```

Методы работы с ячейками таблицы

Для объектов типа "ТаблицаЗначений" определены атрибуты:

- НомерСтроки — номер текущей строки таблицы значений;
- ИдетификаторКолонки — значение элемента таблицы в текущей строке и в колонке, заданной идентификатором.

> **Внимание**
>
> В тексте программного модуля используется идентификатор конкретной колонки, который задается при создании колонки при помощи методов НоваяКолонка() и ВставитьКолонку(). Если при создании колонки ее идентификатор не указан, то обращение к колонке возможно только по номеру.

Значения ячеек с помощью атрибутов задается только для текущей строки. Доступ к остальным ячейкам таблицы осуществляется изложенными в данном разделе методами. Характерный пример использования ячейки таблицы:

```
ТаблицаОстатков.СуммаНаСкладах = БухИт.СКД("С");
```

1. `УстановитьЗначение()`.

Метод `УстановитьЗначение(Строка,Колонка,Знач)` позволяет установить значение ячейки таблицы значений.

Параметры:

- `Строка` — номер строки;
- `Колонка` — номер или идентификатор колонки;
- `Знач` — устанавливаемое значение в ячейке таблицы.

2. `ПолучитьЗначение()`.

Метод `ПолучитьЗначение(Строка,Колонка)` позволяет получить значение заданной ячейки таблицы значений.

Возвращает значение заданной ячейки.

Параметры:

- `Строка` — номер строки;
- `Колонка` — номер или идентификатор колонки.

3. `НайтиЗначение()`.

Метод `НайтиЗначение(Знач,Строка,Колонка)` позволяет найти заданное значение в таблице значений.

Возвращает значение:

- 0 — значение не найдено;
- 1 — значение найдено.

Параметры:

- `Знач` — значение для поиска;
- `Строка` — идентификатор переменной, куда возвращается номер найденной строки (если при вызове метода передать в этот параметр номер строки, то поиск будет осуществляться только по указанной строке);
- `Колонка` — номер (идентификатор) колонки, куда возвращается номер найденной колонки или идентификатор переменной, куда возвращается номер найденной колонки (если при вызове метода передать в этот параметр номер или идентификатор колонки, то поиск будет осуществляться только по указанной колонке).

В процедуре формирования извещения использовались методы работы с таблицей значений "ТаблицаТарифов":

- `НоваяКолонка()`;
- `НоваяСтрока()`;
- `НайтиЗначение()`;
- `ПолучитьСтрокуПоНомеру()`.

Для работы с таблицей значений "Расшифровка" использовались методы:

- `НоваяКолонка()`;
- `НоваяСтрока()`;
- `ВыбратьСтроки()`;
- `ПолучитьСтроку()`.

Методы работы со всей таблицей значений

1. `Сортировать()`.

Метод `Сортировать(Колонки,ДокумПоДате)` позволяет сортировать таблицу значений по колонкам.

Параметры:

- `Колонки` — строковое выражение, которое определяет колонки, порядок и направление сортировки (знак направления сортировки можно указывать до или после обозначения колонки через пробел или без пробела, по умолчанию направление сортировки принимается по возрастанию);

Внимание

Формат передаваемой строки — это разделенные запятыми номера или идентификаторы колонок со знаком направления сортировки: плюс — сортировать по возрастанию; минус — сортировать по убыванию; звездочка — сортировать по внутреннему значению).

- `ДокумПоДате` — число (необязательный параметр), имеет смысл только в том случае, если значениями таблицы значений являются документы, в этом случае можно задавать сортировку документов по их хронологии:
  - 1 — сортировка по хронологии документов;
  - 0 — нет (значение по умолчанию — 0).

2. `Итог()`.

Метод `Итог(Колонка)` позволяет вычислить сумму по колонке таблицы значений.

Параметр — `Колонка` определяет номер или идентификатор колонки, по которой считать сумму.

3. `Заполнить()`.

Метод `Заполнить(Знач,НачСтрока,КонСтрока,Колонки)` позволяет заполнить соответствующие ячейки таблицы значений переданным значением.

Параметры:

- `Знач` — значение одиночное, список значений или таблица значений;
- `НачСтрока` — номер начальной строки (необязательный параметр), с которой надо начинать заполнение (значение по умолчанию — 1);
- `КонСтрока` — номер последней строки (необязательный параметр), по которую надо заполнять (если параметр не указан, то заполнение идет до последней строки);
- `Колонки` — номера или идентификаторы колонок (необязательный параметр), которые надо заполнять (если параметр не задан, то заполняются все колонки).

4. `Очистить()`.

Метод `Очистить()` позволяет очистить таблицу значений и удалить колонки.

5. `КоличествоСтрок()`.

Метод `КоличествоСтрок(Колич)` устанавливает (возвращает) количество строк в таблице значений.

Параметр `Колич` определяет новое количество строк в таблице значений (необязательный параметр).

6. `Свернуть()`.

Метод `Свернуть(ГруппКолонки,СуммКолонки)` позволяет свернуть таблицу значений по соответствующим значениям колонок, т. е. заменяет на одну строку все дублирующие (по значениям группировочных колонок) строки, суммируя значения по заданным колонкам.

Параметры:

- `ГруппКолонки` — группировочные колонки (номера или идентификаторы колонок через запятую), по которым надо группировать данные;
- `СуммКолонки` — суммируемые колонки (номера или идентификаторы колонок через запятую), по которым надо суммировать данные.

7. `Выгрузить()`.

Метод `Выгрузить(Знач,НачСтрока,КонСтрока,Колонки)` позволяет выгрузить соответствующие ячейки таблицы значений.

Параметры:

- `Знач` — значение типа "Таблица значений" или "Список значений", в которое нужно выгрузить данные (если переданное значение пустое, тогда система сама создаст объект типа "Таблица значений");
- `НачСтрока` — номер начальной строки (необязательный параметр), с которой надо начинать выгрузку (значение по умолчанию — 1);
- `КонСтрок` — номер последней строки (необязательный параметр), по которую надо выгружать (если параметр не указан, то выгрузка происходит до последней строки);
- `Колонки` — номера или идентификаторы колонок (необязательный параметр), которые надо выгружать (если параметр не задан, то выгружаются все колонки).

8. `Загрузить()`.

Метод `Загрузить(ТаблицаЗначений)` позволяет скопировать структуру и значения таблицы значений. Прежняя структура колонок таблицы значений при этом очищается.

Параметры:

- `ТаблицаЗначений` — значение типа "Таблица значений", определяющее структуру и значения для загрузки.

9. `ТекущаяСтрока()`.

Метод `ТекущаяСтрока(Строка)` позволяет установить (определить) текущую строку таблицы в элементе диалога типа "ТаблицаЗначений". Возвращает число, соответствующее индексу текущей строки поля диалога (до его изменения), или 0, если текущей строки нет.

Параметры:

- `Строка` — числовое выражение с задаваемым индексом строки для элемента диалога типа "ТаблицаЗначений", на которую требуется установить курсор (необязательный параметр, если параметр не задан, то положение курсора в поле диалога не меняется).

10. `ТекущаяКолонка()`.

Метод `ТекущаяКолонка(НоваяКолонка,ТекущаяКолонка)` позволяет установить (определить) текущую колонку таблицы в элементе диалога типа "ТаблицаЗначений". Возвращает код текущей колонки поля диалога (до его изменения) или "" — пустая строка, если текущей колонки нет.

Параметры:

- `НоваяКолонка` — номер или код колонки для элемента диалога типа "ТаблицаЗначений", на которую требуется установить курсор (необязательный параметр, если параметр не задан, то текущая колонка в поле диалога не меняется);
- `ТекущаяКолонка` — идентификатор переменной, куда система возвращает номер текущей колонки (необязательный параметр).

Внимание

Данный метод можно использовать только для объектов, которые созданы при помощи визуальных средств конфигуратора. В форму вставлены элементы диалога "ТаблицаЗначений", а идентификаторы этих элементов доступны в контексте программного модуля этой формы как уже существующие объекты типа "ТаблицаЗначений".

11. `ВидимостьКолонки()`.

Метод `ВидимостьКолонки(Колонки,Видимость,Позиция)` позволяет показать (скрыть) колонки таблицы значений в визуальном представлении таблицы значений.

Параметры:

- `Колонки` — строковое выражение, которое определяет список колонок (формат передаваемой строки определен как разделенные запятыми номера или идентификаторы колонок, для которых применяется данный метод);
- `Видимость` — число (необязательный параметр), принимающее значения:
  - 1 — показать колонки;
  - 0 — скрыть колонки (значение по умолчанию — 1);
- `Позиция` — позиция, в которой показывать колонку (необязательный параметр, если параметр не задан, то колонки отображаются в соответствии с их порядком в таблице).

12. `Фиксировать()`.

Метод `Фиксировать(КолСтрок,КолКолонок)` позволяет фиксировать в элементе диалога типа "ТаблицаЗначений" колонки и строки.

Параметры:

- `КолСтрок` — количество фиксируемых строк (необязательный параметр, если параметр не указан, то не изменять текущую фиксацию);
- `КолКолонок` количество фиксируемых колонок (необязательный параметр, если параметр не указан, то не изменять текущую фиксацию).

13. `ВыводитьПиктограммы()`.

Метод `ВыводитьПиктограммы(Колонка,Пиктограмма)` позволяет выводить в элементе диалога типа "ТаблицаЗначений" пиктограммы.

Параметры:

- `Колонка` — номер или идентификатор колонки, которая содержит номера пиктограмм;
- `Пиктограмма` — начальный номер пиктограммы (необязательный параметр, по умолчанию — 1).

Внимание

Метод устанавливает режим, при котором в колонке выводится не текст, а пиктограмма. Пиктограмма будет браться из картинки, назначенной элементу диалога "ТаблицаЗначений" в закладке **Картинка** в конфигураторе. Картинка должна быть формата — bmp, содержать все пиктограммы для этой таблицы значений и состоять из последовательности пиктограмм размером 16×15 пикселов. Пиктограммы будут выбираться из картинки по номеру, взятому из числового значения данной колонки в текущей строке.

Для примера рассмотрим фрагмент программного кода, поясняющего механизм получения итоговой таблицы количества оборудования на всех складах:

```
ВыгрузитьТабличнуюЧасть(ТаблицаОстатков,
                                    "Оборудование, Количество");
ТаблицаОстатков.Свернуть("Оборудование", "Количество");
```

Методы таблиц

В разделе "Развернутое сальдо по субконто" была приведена процедура формирования извещения, которая вызывается из журнала расчетов "Начисления" нажатием кнопки  (см. рис. 3.28).

Шаблон таблицы используется из текущей формы **Таблица** (рис. 3.29).

Ссылки на используемые в данной процедуре методы будут сделаны по ходу их рассмотрения.

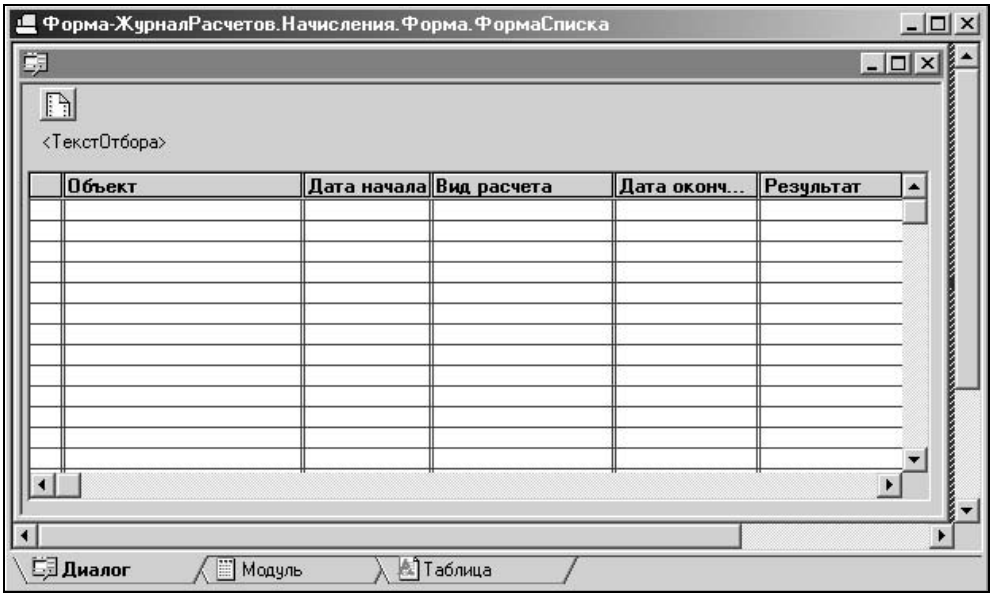


Рис. 3.28. Диалоговое окно формы журнала расчетов "Начисления" на вкладке **Диалог**

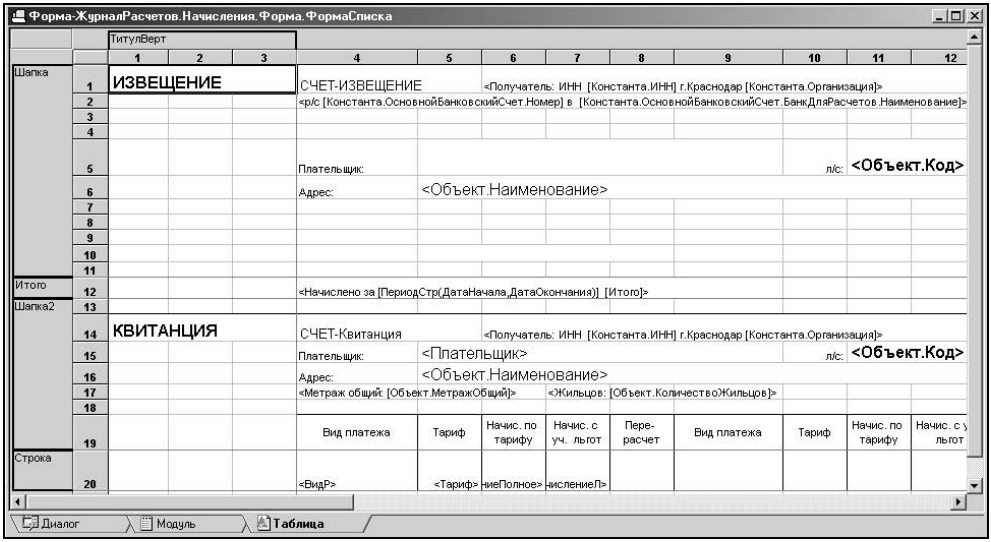


Рис. 3.29. Диалоговое окно формы журнала расчетов "Начисления" на вкладке **Таблица**

1. `ТолькоПросмотр()`.

Метод `ТолькоПросмотр(Флаг)` позволяет установить флаг режима редактирования таблицы. Возвращает текущее числовое значение режима редактирования таблицы (на момент до исполнения метода).

Параметры:

- `Флаг` — число, определяющее режим редактирования:
  - 1 — только просмотр;
  - 0 — допускается редактирование.

В процедуре формирования извещения данный метод используется в виде:

```
Таб.ТолькоПросмотр(1);
```

2. `Показать()`.

Метод `Показать(Заголовок,ИмяФайла,Активизировать)` позволяет открыть окно редактирования таблицы.

Параметры:

- `Заголовок` — заголовок окна редактирования (необязательный параметр, если параметр не задан, в заголовке будет выдаваться слово "Таблица");
- `ИмяФайла` — имя файла для сохранения таблицы (необязательный параметр), причем если параметр задан, то при закрытии окна табличного документа система будет предлагать сохранить документ в файле с указанным именем (если файла с таким именем не существует, то будет создан новый файл с таким именем. Если параметр опущен или имеет пустое значение, то при закрытии окна табличного документа система не будет предлагать сохранить данные в файле);
- `Активизировать` — число, определяющее признак активизации (необязательный параметр):
  - 1 — активизировать окно табличного документа;
  - 0 — не активизировать окно табличного документа;
  - −1 — закрыть окно, если оно открыто (по умолчанию — 1).

В процедуре формирования извещения данный метод используется в виде:

```
Таб.Показать("Извещение","");
```

3. `Очистить()`.

Метод `Очистить()` очищает содержимое табличного документа, уже открытого в окне с помощью метода `Показать()`. Использование данного метода имеет смысл, если сам объект типа "Таблица" не уничтожался после первого заполнения.

4. `Защита()`.

Метод `Защита(Флаг)` позволяет защитить таблицу от изменений (редактирования и копирования, в том числе через буфер обмена). Возвращает текущее числовое значение флага защиты таблицы на момент до исполнения метода.

Параметры:

- `Флаг` — число, определяющее флаг защиты таблицы от изменений и принимающее значение:
  - 1 — установить защиту;
  - 0 — снять.

5. `Записать()`.

Метод `Записать(ИмяФайла,ТипФайла)` позволяет записать таблицу в файл.

Параметры:

- `ИмяФайла` — имя файла;
- `ТипФайла` — числовое или строковое выражение (необязательный параметр), определяющее тип файла:
  - отсутствует, 0 или тип MXL — формат 1С;
  - 1 или тип XLS — формат Ms Excel;
  - 2, тип HTM или HTML — формат HTML;
  - 3 или тип TXT — формат текстовый.

Внимание

Метод может использоваться при работе с таблицей в режиме ввода данных.

6. `Опции()`.

Метод `Опции (ВыводСетки, ВыводЗаголовков, ФиксСтрок, ФиксСтолбцов, ИмяОпцийПечати, ИмяСохрРазмОкна, ФлагЧБПросмотра, НаправлПерехода)` позволяет установить флаги вывода сетки, заголовков, фиксации строк и столбцов, набор опций печати.

Параметры:

- `ВыводСетки` — флаг вывода сетки (необязательный параметр):
  - 1 — показывать;
  - 0 — не показывать;
  - умолчание — 1;
- `ВыводЗаголовков` — флаг вывода заголовков строк и столбцов (необязательный параметр):
  - 1 — показывать,
  - 0 — не показывать (по умолчанию — 1);
- `ФиксСтрок` — число фиксируемых строк (необязательный параметр, по умолчанию — 0);
- `ФиксСтолбцов` — число фиксируемых столбцов (необязательный параметр, по умолчанию — 0);

- ИмяОпцийПечати — строковый идентификатор набора опций печати (необязательный параметр, по умолчанию — пустая строка, в этом случае используются системные опции печати по умолчанию);
- ИмяСохрРазмОкна — строковый идентификатор сохраняемых параметров размера окна (необязательный параметр, по умолчанию — пустая строка, в этом случае размеры окна не запоминаются), причем если этот параметр указан, то система будет сохранять размер окна и использовать его при следующем открытии табличного документа;
- ФлагЧБПросмотра — число (необязательный параметр), принимающее значение:
  - 1 — черно-белый просмотр;
  - 0 — обычный режим просмотра (значение по умолчанию — 0);
- НаправлПерехода — число (необязательный параметр), принимающее значение:
  - 1 — по строкам, т. е. при вводе данных в ячейки, при нажатии клавиши <Enter>, будет автоматически выполняться переход к следующей вводимой ячейке в этой строке, а если таковых нет, то к самой левой вводимой ячейке следующей строки;
  - 2 — по столбцам, т. е. при вводе данных в ячейки, при нажатии клавиши <Enter>, будет автоматически выполняться переход к следующей вводимой ячейке в этом столбце, а если таковых нет, то к самой верхней вводимой ячейке следующего столбца;
  - 3 — при вводе данных в ячейки автоматический переход при нажатии клавиши <Enter> выполняться не будет (значение по умолчанию — 1).

Внимание

Метод может использоваться при работе с таблицей в режиме ввода данных.

В процедуре формирования извещения данный метод используется в виде:

```
Таб.Опции(0,0,0,0,"ОпцииСправки");
```

7. ОбластьПечати().

Метод ОбластьПечати(Верх,Лево,Низ,Право) устанавливает область печати табличного документа.

Параметры:

- Верх — номер верхней печатаемой строки;
- Лево — номер крайнего левого печатаемого столбца;
- Низ — номер нижней печатаемой строки;
- Право — номер крайнего правого печатаемого столбца.

Внимание

Метод может использоваться при работе с таблицей в режиме ввода данных.

8. `ПараметрыСтраницы()`

Метод `ПараметрыСтраницы(Ориентация,Масштаб,РежимПечатиКопий,ПолеСлева, ПолеСправа,ПолеСверху,ПолеСнизу,КолонтитулСверху,КолонтитулСнизу, Автомасштаб,ФлагЧБПечати,ИмяПринтера)` позволяет установить параметры страницы.

Возвращаемого значения нет.

Параметры:

- `Ориентация` — ориентация вывода на печать (необязательный параметр):
  - 1 — портрет;
  - 2 — ландшафт;
- `Масштаб` — числовое выражение, определяющее масштаб (в процентах) вывода на печать (необязательный параметр);
- `РежимПечатиКопий` — числовое выражение, определяющее режим вывода нескольких копий на печать (необязательный параметр):
  - 0 — (collate) выводить сначала первые страницы всех копий, затем вторые и т. д.;
  - 1 — (разобрать) выводить страницы копий по порядку;
- `ПолеСлева` — числовое выражение, определяющее расстояние (в миллиметрах) от левого края страницы (необязательный параметр);
- `ПолеСправа` — числовое выражение, определяющее расстояние (в миллиметрах) от правого края страницы (необязательный параметр);
- `ПолеСверху` — числовое выражение, определяющее расстояние (в миллиметрах) от верхнего края страницы (необязательный параметр);
- `ПолеСнизу` — числовое выражение, определяющее расстояние (в миллиметрах) от нижнего края страницы (необязательный параметр);
- `КолонтитулСверху` — числовое выражение, определяющее размер (в миллиметрах) верхнего колонтитула (необязательный параметр);
- `КолонтитулСнизу` — числовое выражение, опредляющее размер (в миллиметрах) нижнего колонтитула (необязательный параметр);
- `Автомасштаб` — режим автоматического подбора масштаба для размещения документа при печати на листе по ширине (необязательный параметр), принимающий значение:
  - 1 — включить;
  - 0 — выключить (по умолчанию — 0);

- ФлагЧБПечати — число (необязательный параметр), принимающее значения:
  - 1 — черно-белая печать;
  - 0 — обычный режим печати (значение по умолчанию − 0);
- ИмяПринтера — необязательный параметр (строка с именем принтера, как в стандартном диалоге печати Windows).

Метод может использоваться при работе с таблицей в режиме ввода данных.

9. КоличествоЭкземпляров().

Метод КоличествоЭкземпляров(Колво) позволяет определить количество печатаемых экземпляров. Возвращает: текущее числовое значение количества печатаемых экземпляров (на момент до исполнения метода).

Параметры — Колво определяет число печатаемых экземпляров.

Метод может использоваться при работе с таблицей в режиме ввода данных.

10. ЭкземпляровНаСтранице().

Метод ЭкземпляровНаСтранице(Колво) позволяет определить количество печатаемых экземпляров на странице. Возвращает текущее числовое значение количества печатаемых экземпляров на странице (на момент до исполнения метода).

Параметры:

- Колво — число печатаемых экземпляров на странице. Может принимать значения:
  - 1 — один экземпляр на странице;
  - 2 — два экземпляра на странице;
  - 0 — автоматический режим размещения двух экземпляров на странице, исходя из размеров документа.

Метод может использоваться при работе с таблицей в режиме ввода данных.

11. Напечатать().

Метод Напечатать(Флаг) позволяет напечатать таблицу без предварительного просмотра (печать без открытия окна редактирования).

Параметры:

- ❒ `Флаг` — режим запроса диалога печати (необязательный параметр), принимающий значение:
  - 1 — запрашивать диалог печати (по умолчанию);
  - 0 — не запрашивать.

Внимание

Метод может использоваться при работе с таблицей в режиме ввода данных.

12. `ЗначениеТекущейЯчейки()`.

Метод `ЗначениеТекущейЯчейки(Адрес)` возвращает вычисленное значение текущей ячейки таблицы (задается в конфигураторе **Свойства ячейки | Текст | Расшифровка**), перенесенное в табличный документ.

Параметры:

- ❒ **Адрес** — идентификатор переменной, куда система возвратит адрес текущей ячейки в формате RnCn" (необязательный параметр).

Методы работы с шаблоном

1. `ИсходнаяТаблица()`.

Метод `ИсходнаяТаблица(Строка)` переназначает в качестве исходной таблицы-шаблона одну из таблиц той формы, в программном модуле которой запущена данная процедура. Имя таблицы сначала ищется в форме модуля, потом в общих таблицах. Если такой таблицы нет, то переданное имя будет рассматриваться как имя файла, содержащего данную таблицу.

Параметр — `Строка` задает имя таблицы формы или имя файла, содержащего таблицу.

Характерный пример использования данного метода в документе "Платежное поручение":

```
Таб = СоздатьОбъект("Таблица");
ИмяФайлаПечатнойФормы = КаталогИБ()+"ExtForms\PrnForms\1cbpp03.mxl";
Если ФС.СуществуетФайл(ИмяФайлаПечатнойФормы) = 1 Тогда
    Таб.ИсходнаяТаблица(ИмяФайлаПечатнойФормы);
Иначе
    Таб.ИсходнаяТаблица("ПлПор");
КонецЕсли;
```

2. `ИспользоватьФормат()`.

Метод `ИспользоватьФормат(СтрокаФормата)` устанавливает формат по умолчанию для вывода выражений секций таблицы.

Возвращает строковое значение, содержащее текущую форматную строку (строку задания формата) по умолчанию для таблицы (на момент до исполнения метода).

Параметры:

- `СтрокаФормата` — строковое выражение, содержащее форматную строку (необязательный параметр, см. метод Формат).

Внимание

В ячейках таблицы, при выводе которых требуется формат, отличный от установленного данным методом, должен быть установлен формат явным образом. Форматная строка записывается через символ "#" после выражения, заданного для ячейки. Если выражение, заданное для ячейки просто завершается символом "#", то будет использоваться системный формат по умолчанию.

Характерный пример использования данного метода в отчете "Книга покупок":

```
Таб.ИсходнаяТаблица("КнигаПокупок");
Таб.ИспользоватьФормат("Ч-15.2-");
```

3. `Открыть()`.

Метод `Открыть(ИмяФайла)` открывает таблицу из файла.

Параметр — `ИмяФайла` представляет строковое выражение с именем файла.

4. `Вывести()`.

Метод `Вывести()` позволяет перенести всю назначенную исходную таблицу-шаблон в результирующую таблицу.

Приведем фрагмент из обработки "Регламентированные отчеты":

```
ВыбОтчет =
 ГруппыОтчетов.ПолучитьЗначение(ГруппыОтчетов.ТекущаяСтрока(),
           ИдГруппыОтчетов)+"\"+ФайлОтчета;
Расширение = ВРег(Прав(ФайлОтчета, 3));
Если Расширение = "ERT" Тогда
   ОткрытьФорму("Отчет",,ВыбОтчет);
// Если выбрана таблица
ИначеЕсли Расширение = "MXL" Тогда
   Таб = СоздатьОбъект("Таблица");
// Откроем таблицу для нашего объекта
   Таб.Открыть(ВыбОтчет);
   Таб.ТолькоПросмотр(1);
   Таб.Опции(0, 0, 0, 0);
   Таб.Показать(НазвОтчета);
ИначеЕсли Расширение = "TXT" Тогда
```

```
    Текст = СоздатьОбъект("Текст");
    Текст.Открыть(ВыбОтчет);
    Текст.Показать(НазвОтчета, "");
КонецЕсли;
```

Методы работы с секцией

Практически во всех методах работы с секцией таблицы задан параметр ИмяСекции. Этот параметр представляет выражение, задающее секцию. Имя секции задается строковым выражением следующего формата:

ИдентификаторСекции1 [< | > | -] [|ИдентификаторСекции2 [< | > | -]].

Символы — "<", ">", "-", после идентификатора секции, указывают на то, что выбирается только часть секции, квадратные скобки — факультативную (необязятельную) часть секции. Причем:

- "<" — заголовочная часть (с начала секции до начала вложенной секции);
- ">" — подвальная часть (с конца вложенной секции до конца секции);
- "-" — средняя часть (собственно вложенная секция).

В выражении можно задавать имена двух секций, разделенных символом — "|". В разультате будет получена область исходной таблицы, являющаяся пересечением первой и второй указанных секций. При этом одна секция может быть горизонтальной (состоять из строк), а другая — вертикальной (состоять из колонок). В результате получится прямоугольная область таблицы.

1. ПолучитьСекцию().

Метод ПолучитьСекцию(ИмяСекции) возвращает именованную секцию исходной таблицы-шаблона. При получении секции, ячейки секции, имеющие тип "Шаблон" и "Выражение", будут заполнены соответствующими данными.

Параметр ИмяСекции определяет выражение, задающее секцию.

2. ВывестиСекцию().

Метод ВывестиСекцию(ИмяСекции) позволяет перенести секцию исходной таблицы-шаблона в результирующую таблицу.

Параметр — ИмяСекции определяет выражение типа строка, задающее имя выводимой секции, или значение типа секция, полученное при помощи метода ПолучитьСекцию().

В процедуре формирования извещения данный метод используется в виде:

```
Таб.ВывестиСекцию("Строка");
```

3. ПрисоединитьСекцию().

Метод ПрисоединитьСекцию(ИмяСекции) позволяет присоединить секцию исходной таблицы-шаблона к текущей таблице.

Параметр — ИмяСекции определяет выражение типа строка, задающее имя присоединяемой секции, или значение типа секция, полученное при помощи метода ПолучитьСекцию().

4. ШиринаСекции().

Метод ШиринаСекции(ИмяСекции) возвращает число столбцов в секции таблицы-шаблона.

Параметры — ИмяСекции определяет выражение типа строка, задающее имя выводимой секции.

5. ВысотаСекции().

Метод ВысотаСекции(ИмяСекции) возвращает число строк в секции таблицы-шаблона.

Параметр — ИмяСекции определяет выражение типа строка, задающее имя выводимой секции.

6. Область().

Метод Область(Адрес) возвращает значение типа "ОбластьТаблицы" области секции таблицы или секции таблицы в режиме ввода данных.

Параметр — Адрес определяется строковым выражением, задающим имя области или адрес в формате "R1C1:R2C2", причем если метод вызван без параметров, то область задана всей секцией (необязательный параметр).

В процедуре формирования извещения данный метод используется в виде:

```
Таб.Область("Долг").Текст = "Долг на "+ ДатаНачала +
                                          ":  "+ Сальдо;
```

7. Область().

Метод Область(R1,C1,R2,C2) возвращает значение типа "ОбластьТаблицы" области выходной секции таблицы или секции таблицы в режиме ввода данных.

Параметры:

- R1 — номер первой строки области (необязательный параметр);
- C1 — номер первого столбца области (необязательный параметр);
- R2 — номер последней строки области (необязательный параметр);
- C2 — номер последнего столбца области (необязательный параметр).

Внимание

Если последняя строка и последний столбец отсутствуют, то область задана единственной ячейкой. Если строки или столбцы отсутствуют, то область задана диапазоном столбцов или строк соответственно. Если метод вызван без параметров, то область задана всей секцией.

В отчете "Карточка субконт" определена процедура:

```
Процедура ВывестиОбороты()
   Период = "Обороты за ";
   Если ВидПериода = 4 Тогда
     Период = Период+НачалоПериода;
   Иначе
      Период = Период+ПериодСтр(НачалоПериода, КонецПериода);
   КонецЕсли;
   Секция = Т.ПолучитьСекцию("Секция_17");
   Если (Валютный = 1) И (ПоВалюте = 1) Тогда
      Т.ВывестиСекцию(Секция);
      Секция = Т.ПолучитьСекцию("Секция_18");
   КонецЕсли;
   Если Количественный = 1 Тогда
      Т.ВывестиСекцию(Секция);
      Секция = Т.ПолучитьСекцию("Секция_19");
   КонецЕсли;
   Область = Секция.Область(1, 1, 1, 9);
   Область.РамкаСнизу(4);
   Т.ВывестиСекцию(Секция);
КонецПроцедуры
```

Внимание

Для таблицы существует одноименный метод, функционально оба метода похожи, но их использование отличается.

Атрибуты и методы объекта типа "ОбластьТаблицы"

Для объектов типа "ОбластьТаблицы" определен атрибут — Текст. Этот атрибут позволяет прочитать или установить значение текста области (аналогично тому, как в конфигураторе интерактивно задают значение формулы в свойствах ячейки таблицы (**Свойства** на закладке **Текст**).

В процедуре формирования таблицы на печать документа "Платежное поручение" используется данный метод. Соответствующая область выделена на рис. 3.30.

```
Таб.Область("Плательщик").Текст = Плательщик;
```

1. `Расшифровка()`.

Метод `Расшифровка(Значение,РежимИспользования)` позволяет установить расшифровку области. Возвращает текущее значение расшифровки.

Параметры:

- `Значение` — новое значение расшифровки области (необязательный параметр);
- `РежимИспользования` — число (необязательный параметр), принимающее значение:
  - 0 — обычный режим;
  - 1 — установить данную расшифровку для всей строки;
  - 2 — не вызывать расшифровку для данной ячейки (значение по умолчанию — 0).

Замечание

Данный метод нельзя использовать при работе с областью таблицы в режиме ввода данных.

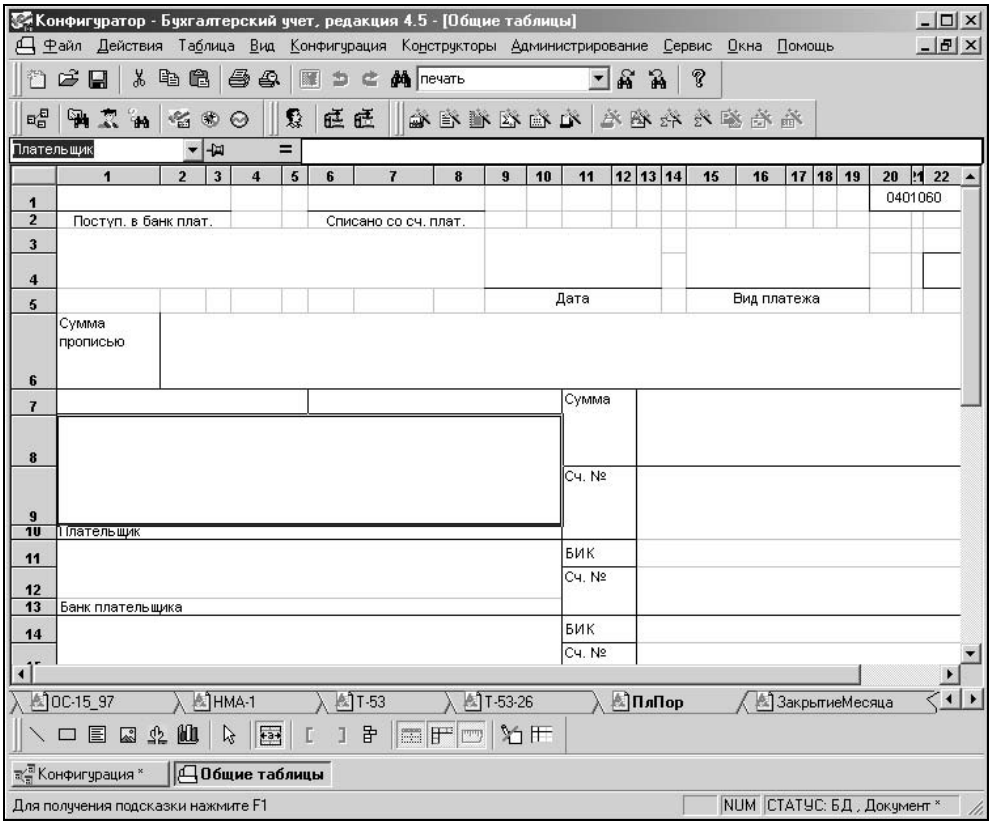


Рис. 3.30. Шаблон таблицы **ПлПор**

2. `Объединить()`.

Метод `Объединить()` позволяет объединить ячейки области.

Внимание

Данный метод нельзя использовать при работе с областью таблицы в режиме ввода данных.

3. `Шрифт()`.

Метод `Шрифт(ИмяШрифта)` позволяет получить или установить шрифт области. Возвращает имя шрифта до исполнения метода.

Параметры:

- `ИмяШрифта` — строковое выражение, задающее имя шрифта или "имя, набор символов", где указываются основные наборы символов (необязательный параметр):
  - `204` — русский;
  - `238` — европейский;
  - `186` — балтийский;
  - `161` — греческий;
  - `162` — турецкий.

Если параметр опущен, то шрифт области не изменяется.

4. `РазмерШрифта()`.

Метод `РазмерШрифта(Размер)` позволяет получить или установить размер шрифта области. Возвращает размер шрифта до исполнения метода.

Параметр — `Размер` представляет числовое выражение, задающее размер шрифта (необязательный параметр), причем если параметр опущен, то размер шрифта области не изменяется.

5. `Полужирный()`.

Метод `Полужирный(Жирный)` позволяет получить или установить признак жирного шрифта области. Возвращает признак жирного шрифта до исполнения метода.

Параметры:

- `Жирный` — число, принимающее значение (необязательный параметр):
  - 1 — жирный шрифт;
  - 0 — не жирный шрифт.

Если параметр опущен, то жирность шрифта области не изменяется.

6. `Курсив()`.

Метод `Курсив(Курсив)` позволяет получить или установить признак шрифта — курсив области. Возвращает признак шрифта курсив до исполнения метода.

Параметры:

- `Курсив` — число, которое может принимать значение (необязательный параметр):
  - 1 — шрифт курсив;
  - 0 — шрифт не курсив.

Если параметр опущен, то признак курсив шрифта области не изменяется.

7. `Подчеркнутый()`.

Метод `Подчеркнутый(Подч)` позволяет получить или установить признак подчеркнутого шрифта области. Возвращает признак подчеркнутого шрифта до исполнения метода.

Параметры:

- `Подч` — число, принимающее значение (необязательный параметр):
  - 1 — шрифт подчеркнутый;
  - 0 — шрифт не подчеркнутый.

Если параметр опущен, то признак подчеркнутого шрифта области не изменяется.

8. `ВертикальноеПоложение()`.

Метод `ВертикальноеПоложение(Положение)` позволяет получить или установить признак вертикального выравнивания текста области. Возвращает признак вертикального выравнивания текста до исполнения метода:

- 1 — установлено вертикальное выравнивание;
- 0 — в противном случае.

Параметры:

- `Положение` — число, которое может принимать значение (необязательный параметр):
  - 1 — выравнивание снизу;
  - 2 — выравнивание сверху;
  - 3 — выравнивание по центру.

Если параметр опущен, то признак вертикального выравнивания текста области не изменяется.

9. `ГоризонтальноеПоложение()`.

Метод `ГоризонтальноеПоложение(Положение)` позволяет получить или установить признак горизонтального выравнивания текста области. Возвращает признак горизонтального выравнивания текста до исполнения метода.

Параметры:

- ❒ `Положение` — число, которое может принимать значение (необязательный параметр):
  - 1 — выравнивание снизу;
  - 2 — выравнивание сверху;
  - 3 — выравнивание по центру.

Если параметр опущен, то признак горизонтального выравнивания текста области не изменяется.

10. `Контроль()`.

Метод `Контроль(Контроль)` позволяет получить или установить признак контроля текста области. Возвращает признак контроля текста до исполнения метода.

Параметры:

- ❒ `Контроль` — число, которое может принимать значение (необязательный параметр):
  - 1 — авто;
  - 2 — обрезать;
  - 3 — забивать;
  - 4 — переносить;
  - 5 — красный;
  - 6 — забивать + Красный.

Если параметр опущен, то признак контроля текста области не изменяется.

В следующей группе методов необязательный параметр `Рамка` имеет следующие значения:

- ❒ 0 — нет;
- ❒ 1 — тонкая точечная;
- ❒ 2 — очень тонкая;
- ❒ 3 — тонкая сплошная;
- ❒ 4 — средняя сплошная;
- ❒ 5 — толстая сплошная;
- ❒ 6 — двойная;
- ❒ 7 — тонкая средний пунктир;
- ❒ 8 — тонкая длинный пунктир;
- ❒ 9 — толстая пунктир.

Если в списке параметров параметр Рамка опущен, то рамка сверху области не изменяется.

Примечание

В процедуре ВвестиОбороты() был использован метод РамкаСнизу().

11. РамкаСверху().

Метод РамкаСверху(Рамка) позволяет получить или установить рамку сверху области. Возвращает рамку сверху до исполнения метода.

12. РамкаСнизу().

Метод РамкаСнизу(Рамка) позволяет получить или установить рамку снизу области. Возвращает рамку снизу до исполнения метода.

13. РамкаСлева().

Метод РамкаСлева(Рамка) получить или установить рамку слева области. Возвращает рамку слева до исполнения метода.

14. РамкаСправа().

Метод РамкаСправа(Рамка) позволяет получить или установить рамку справа области. Возвращает рамку справа до исполнения метода.

15. Рамка().

Метод Рамка(РамкаСлева,РамкаСверху,РамкаСправа,РамкаСнизу) позволяет установить рамки всех ячеек области.

Методы журнала расчетов

Рассмотрим алгоритм формирования журнала расчетов (см. рис. 3.31). Здесь процедура ОбработкаПроведения() представлена кодом, который приведен ниже.

```
Процедура ОбработкаПроведения()
  ЖурналРасчетовНачислений =
                    СоздатьОбъект("ЖурналРасчетов.Начисления");
  ЖурналРасчетовНач = СоздатьОбъект("ЖурналРасчетов.Начисления");
  НТП = ЖурналРасчетовНачислений.НачалоТекущегоПериода();
  КТП = ЖурналРасчетовНачислений.КонецТекущегоПериода();
  Тарифы = СоздатьОбъект("Справочник.Услуги");
  Тарифы.ВыбратьЭлементы();
  Пока Тарифы.ПолучитьЭлемент()=1 Цикл
     Пока ПолучитьСтроку()=1 Цикл
```

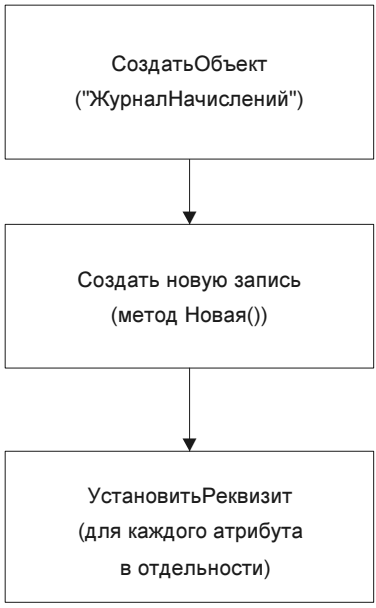


Рис. 3.31. Алгоритм журнала расчетов

```
//Создадим записи в журнале расчетов
Если Тарифы.Расчет.ВходитВГруппу(ГруппаРасчетов.КвартПлата) = 1
        тогда
           Если ((Тарифы.Расчет =
           ВидРасчета.ХолоднаяВодаПоВодомеру)или
           (Тарифы.Расчет = ВидРасчета.ГорячаяВодаПоВодомеру)) и
           (Квартира.ВидУчетаВоды = 1) тогда
              Продолжить;
           Иначе
              Если    ЖурналРасчетовНачислений.ВыбратьПериодПоОбъекту
                        (Квартира,КТП) = 1 Тогда
                 ЖурналРасчетовНачислений.УстановитьРеквизит
                                  ( "НомерСтрокиДокумента",
                                  НомерСтроки);
                 ЖурналРасчетовНачислений.ВвестиРасчет
                              (Квартира,Тарифы.ТекущийЭлемент().Расчет,
                              НТП,КТП);
              Иначе
ЖурналРасчетовНачислений.Новая();
ЖурналРасчетовНачислений.УстановитьРеквизит("Объект",Квартира);
ЖурналРасчетовНачислений.УстановитьРеквизит("ВидРасч",
                                                  Тарифы.Расчет);
```

```
ЖурналРасчетовНачислений.УстановитьРеквизит("Документ",
                                                ТекущийДокумент());
ЖурналРасчетовНачислений.УстановитьРеквизит
                    ("РодительскийДокумент",ТекущийДокумент());
ЖурналРасчетовНачислений.УстановитьРеквизит("ДатаНачала",НТП);
ЖурналРасчетовНачислений.УстановитьРеквизит( "ДатаОкончания",КТП);
ЖурналРасчетовНачислений.УстановитьРеквизит(
                              "НомерСтрокиДокумента", НомерСтроки);
ЖурналРасчетовНачислений.Записать();
            КонецЕсли;
          КонецЕсли;
        КонецЕсли;
      КонецЦикла;
  КонецЦикла;
  Жильцы = СоздатьОбъект("Справочник.Жильцы");
  ТаблицаНачислений = СоздатьОбъект("ТаблицаЗначений");
  ТаблицаНачислений.НоваяКолонка("Контрагент",
                                        "Справочник.Контрагенты");
  ТаблицаНачислений.НоваяКолонка("Сумма");
  ЖурналРасчетовНач.ВыбратьЗаписиПоДокументу(ТекущийДокумент());
  Пока ЖурналРасчетовНач.ПолучитьЗапись()=1 цикл
     ЖурналРасчетовНач.Рассчитать();
     Если ЖурналРасчетовНач.Результат>0 тогда
        Жильцы.ИспользоватьВладельца(ЖурналРасчетовНач.Объект);
        Жильцы.ВыбратьЭлементы();
        Пока Жильцы.ПолучитьЭлемент()=1 цикл
           Если Жильцы.СтатусПрописки=Перечисление.Статусы.Владелец
              тогда
              ТаблицаНачислений.НоваяСтрока();
              ТаблицаНачислений.Контрагент = Жильцы.Контрагент;
              ТаблицаНачислений.Сумма = ЖурналРасчетовНач.Результат;
              Прервать;
           КонецЕсли;
        КонецЦикла;
     КонецЕсли;
  КонецЦикла;
  ТаблицаНачислений.Свернуть("Контрагент","Сумма");
  ТаблицаНачислений.ВыбратьСтроки();
  Пока ТаблицаНачислений.ПолучитьСтроку()=1 Цикл
     Операция.НоваяПроводка();
```

```
    Операция.Кредит.Счет = СчетПоКоду("90.1.1");
    Операция.Кредит.ВидыНоменклатуры =
                              Константа.ОсновнойВидНоменклатуры;
    Операция.Дебет.Счет = СчетПоКоду("62.1");
    Операция.Дебет.Контрагенты = ТаблицаНачислений.Контрагент;
    Операция.Сумма = ТаблицаНачислений.Сумма;
    Операция.СодержаниеПроводки = "Начисл. за ком. услуги";
  КонецЦикла;
  Операция.Записать();
КонецПроцедуры
```

Методы, используемые в процедуре, можно определить следующим образом:

1. `НачалоТекущегоПериода()`.

Метод `НачалоТекущегоПериода()` возвращает дату начала текущего расчетного периода журнала расчетов.

В приведенной процедуре метод использован в строке кода:

```
НТП = ЖурналРасчетовНачислений.НачалоТекущегоПериода();
```

2. `КонецТекущегоПериода()`.

Метод `КонецТекущегоПериода()` возвращает дату окончания текущего расчетного периода журнала расчетов.

В приведенной процедуре метод использован в строке кода:

```
КТП = ЖурналРасчетовНачислений.КонецТекущегоПериода();
```

3. `НачалоПериодаПоДате()`.

Метод `НачалоПериодаПоДате(Дата)` возвращает дату начала произвольного расчетного периода журнала расчетов.

Параметр — `Дата` является любой датой, которая попадает в требуемый период.

4. `КонецПериодаПоДате()`.

Метод `КонецПериодаПоДате(Дата)` возвращает дату окончания произвольного расчетного периода журнала расчетов.

Параметр — `Дата` является любой датой, которая попадает в требуемый период.

5. `ПериодПоДате()`

Метод `ПериодПоДате(Дата)` возвращает период журнала расчетов по дате.

Параметр — `Дата` представляет значение типа "дата".

6. `УстановитьТекущийПериод()`.

Метод `УстановитьТекущийПериод(Период,Способ)` устанавливает текущий период журнала расчетов. Возвращает число:

- 1 — выполнено;
- 0 — не выполнено.

Параметры:

- ❒ Период — значение типа "Период журнала расчетов";
- ❒ Способ — число (необязательный параметр), принимающее значение:
  - 0 — не отрабатывать системные действия, связанные со сменой периода;
  - 1 — отрабатывать системные процедуры (например, отменить расчет записей при откате "назад" или провести архивацию документов при смене периода "вперед"), причем в этом режиме метод ведет себя как интерактивная смена периода, но без вопросов (значение параметра по умолчанию — 1).

7. ТекущийПериод().

Метод ТекущийПериод() возвращает значение типа "Период журнала расчетов".

Методы констант

Методы периодических реквизитов

1. Получить().

Метод Получить(Дата) возвращает значение периодической константы на заданную дату.

Параметр — Дата — необязательный параметр, определяющийся выражением типа "Дата" или значением типа "Документ" или "Позиция документа". Этот параметр задает момент времени, на который требуется получить значение периодической константы. Значение по умолчанию — точка актуальности ТА, если используется компонента "Оперативный учет", Рабочая дата — если компонента "Оперативный учет" не используется.

Создадим в нашей конфигурации отчет "Должники". Для формирования отчета используется периодическая константа ПределДолга. Перед формированием списка должников получим предел долга на заданный период:

```
Константа.ПределДолга.Получить(КонМесяца(Дата);
```

2. Установить().

Метод Установить(Дата,Значение) позволяет установить значение периодической константы на дату.

Параметры:

- ❒ Дата — дата, на которую требуется установить значение периодической константы;
- ❒ Значение — новое значение константы.

Метод часто используется в формах настройки пользовательских установок, в которых настраиваются периодические константы.

Глава 4

Интерфейс с другими программными продуктами

Взаимодействие разных программ осуществляется чаще всего обменом информацией на уровне файлов. Из одной программы информация выгружается в файл определенной структуры и затем, в виде этого файла, загружается в другую. В большинстве программ предусмотрен режим экспорта-импорта данных. Для конфигураций "1С:Предприятие" такой экспорт предусмотрен для выгрузки данных, для предоставления в ИМНС (Инспекция Министерства Налогов и Сборов) и Пенсионный фонд, а также для обмена данными между различными конфигурациями системы. Для организации такого взаимодействия необходимо владеть методами работы как с файловой системой, так и с базой данных.

Рассмотрим организацию обмена данными между конфигурацией "1С:Предприятие.Зарплата и кадры", с одной стороны, и программой "ZAVOD", которая осуществляет передачу данных в банк для расчетов с сотрудниками, посредством пластиковой карты, с другой. Программа "ZAVOD" предусматривает импорт данных из файла import.dbf в кодировке DOS (рис. 4.1).

IMPORT : таблица

	TAB_N	NAME	SERNUM	PDATE	PWHR	PREM	OTDEL	SUM_PAY	TEL	TEX
▶	1	Иванов Иван Иванович	ХХХ-АГ/123456	02.02.2004	Прикубанским	1	Отдел1	1100,00		
*								0,00		

Запись: 1 из 1

Рис. 4.1. Диалоговое окно таблицы IMPORT в программе ACCESS

В конфигурации "1С:Преприятие.Зарплата и кадры" выплата заработной платы производится посредством документа "Выплата заработной платы". Модифицируем документ дополнительной кнопкой **Выгрузить** с назначенной процедурой `Выгрузить()` (рис. 4.2). Для задания признака выплаты через пластиковую карту, в справочнике "Сотрудники" зададим реквизит "ЧерезКарту".

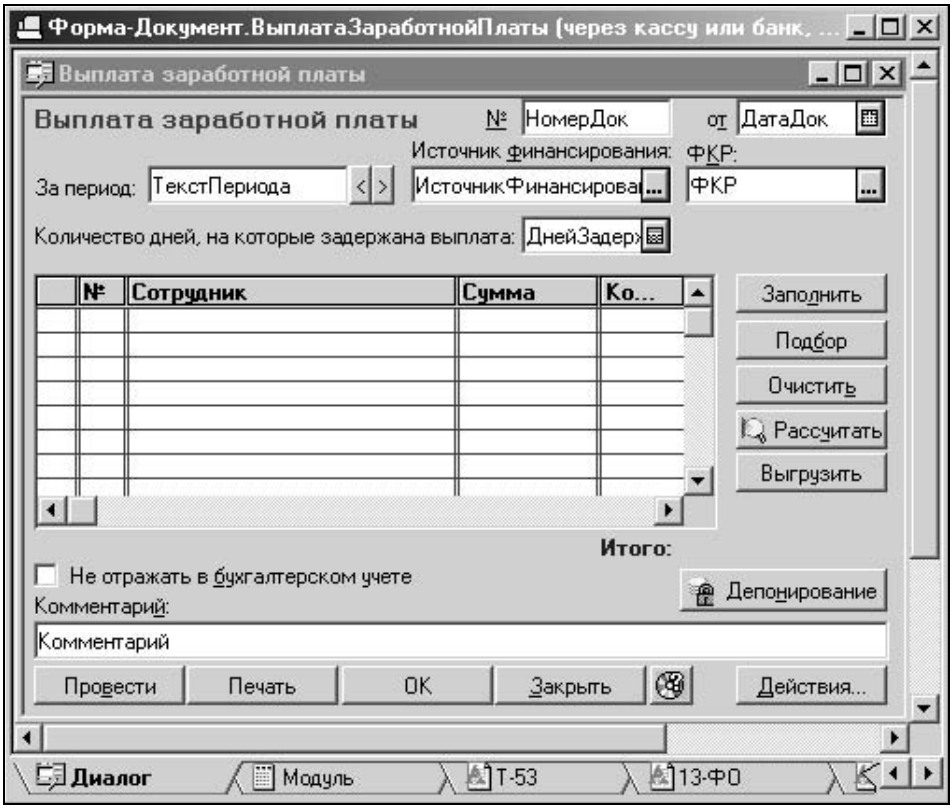


Рис. 4.2. Диалоговое окно
модифицированного документа "Выплата заработной платы"

После заполнения и расчета таблицы документа, нажатием кнопки **Выгрузить**, выполнится следующая процедура:

```
Процедура Выгрузить()
  БД=СоздатьОбъект("XBase");
  БД.КодоваяСтраница(1);
  ИмяФайла="";
  Каталог="";
  Наименование="";
  ФС.ВыбратьФайл(0,ИмяФайла,Каталог,"Выберите файл выгрузки
                """+Наименование+""""",
                "Файлы прописи(*.dbf)|*.dbf");
  БД.ОткрытьФайл(Каталог+ИмяФайла,"",0);
  БД.КодоваяСтраница(1);
  БД.ОчиститьФайл();
  ВыбратьСтроки();
```

```
  Пока ПолучитьСтроку()=1 Цикл
     Если Сотрудник.ОплатаКартой=1 тогда
        БД.Добавить();
        БД.TAB_N = Прав(Сотрудник.Код,8);
        БД.NAME = Сотрудник.Наименование;
        СерияНомерДокумента = "";
        ВидДокумента = "";
        Документ = глРазложитьДокУдостоверяющийЛичность
                                    ( Сотрудник.ДокументУдЛичность);
        Если Документ.РазмерСписка() > 4 Тогда
          СпрДокумент = СоздатьОбъект( "Справочник.
                       ДокументыУдостоверяющиеЛичность");
          СпрДокумент.НайтиПоНаименованию
                       (Документ.ПолучитьЗначение(1));
          Вид = СпрДокумент.ТекущийЭлемент().КодСЗВ;
          Если СокрЛП(Вид)="паспорт" тогда
             БД.PASSCODE = 1;
             СерияНомерДокумента =
             ВРег(?(ПустоеЗначение(Документ.ПолучитьЗначение(2))=1,
             "", Документ.ПолучитьЗначение(2)))+"/
             "+ Документ.ПолучитьЗначение(3)
          ИначеЕсли СокрЛП(Вид)="паспорт россии" тогда
             БД.PASSCODE = 8;
             СерияНомерДокумента =
             ВРег(?(ПустоеЗначение(Документ.ПолучитьЗначение(2))=1,
             "", Документ.ПолучитьЗначение(2)))+"
             "+ Документ.ПолучитьЗначение(3);
          Иначе
             Продолжить;
          КонецЕсли;
          БД.SERNUM = СерияНомерДокумента;
          БД.PDATE = Документ.ПолучитьЗначение(4);
          БД.PWHR = Документ.ПолучитьЗначение(5);
        КонецЕсли;
        БД.OTDEL = Сотрудник.
        Подразделение.Получить(РабочаяДата()).Наименование;
        БД.SUM_PAY = Сумма;
        БД.Записать();
     КонецЕсли;
  КонецЦикла;
  БД.ЗакрытьФайл();
КонецПроцедуры  // Выгрузить()
```

Методы файловой системы

При использовании методов файловой системы следует обратить особое внимание на то обстоятельство, что используется объект типа "ФС" и создавать указатель на него с помощью метода `СоздатьОбъект()` не надо. Однако в модуле формы отчета "Регламентированные отчеты" определена функция проверки каталога `ПустойКаталог()`, в которой такой указатель создается:

```
Функция ПустойКаталог(ТекКаталог)
  КаталогПуст = 1;
  ПромКаталог = СокрЛП(ТекКаталог);
  Если Прав(ПромКаталог, 1) <> "\" Тогда
     ПромКаталог = ПромКаталог + "\";
  КонецЕсли;
// проверяем, является ли найденный каталог пустым
  ОФС = СоздатьОбъект("ФС");
  ИФ = ОФС.НайтиПервыйФайл(ПромКаталог + "*.*");
  Пока ПустаяСтрока(ИФ) = 0 Цикл
     Если (ИФ <> ".") И (ИФ <> "..") Тогда
        КаталогПуст = 0;
        Прервать;
     КонецЕсли;
     ИФ = ОФС.НайтиСледующийФайл();
  КонецЦикла;
  ОФС = "";
  Возврат КаталогПуст;
КонецФункции
```

Теперь рассмотрим методы, которые используются для работы с объектами файловой системы.

1. `ВыбратьФайл()`.

Метод `ВыбратьФайл(ТипДиалога,ИмяФайла,ИмяНачКаталога,ЗаголовокОкна, Фильтр,Расширение,Таймаут)` открывает окно диалога выбора или сохранения файла.

Возвращает:

- 0 — если в окне диалога нажата кнопка **Отмена**;
- 1 — если нажата кнопка **ОК**.

Параметры:

- `ТипДиалога`:
  - 0 — диалог типа "открыть";
  - 1 — диалог типа "сохранить";

- `ИмяФайла` — переменная, содержащая на входе строку с именем файла, а на выходе имя выбранного файла;
- `ИмяНачКаталога` — переменная, содержащая на входе строку с именем начального каталога, а на выходе имя выбранного каталога;
- `ЗаголовокОкна` — строка с заголовком окна;
- `Фильтр` — строка с фильтром отбора файлов (например, все файлы — *.*);
- `Расширение` — строка с расширением файла по умолчанию;
- `Таймаут` — время ожидания отклика пользователя в секундах (необязательный параметр).

В процедуре выгрузить используется данный метод в фрагменте:

```
ИмяФайла="";
Каталог="";
Наименование="";
ФС.ВыбратьФайл(0,ИмяФайла,Каталог,"Выберите файл выгрузки
                """+Наименование+""""","Файлы прописи(*.dbf)|*.dbf");
```

2. `ВыбратьФайлКартинки()`.

Метод `ВыбратьФайлКартинки(ТипДиалога,ИмяФайла,ИмяНачКаталога, ЗаголовокОкна,Расширение,Таймаут)` открывает окно диалога выбора или сохранения файла картинки.

Возвращает:

- 0 — если в окне диалога нажата кнопка **Отмена**;
- 1 — если нажата кнопка **ОК**.

Параметры:

- `ТипДиалога`:
  - 0 — диалог типа "открыть";
  - 1 — диалог типа "сохранить";
- `ИмяФайла` — переменная, содержащая на входе строку с именем файла, а на выходе имя выбранного файла;
- `ИмяНачКаталога` — переменная, содержащая на входе строку с именем начального каталога, а на выходе имя выбранного каталога;
- `ЗаголовокОкна` — строка с заголовком окна;
- `Расширение` — строка с расширением файла по умолчанию;
- `Таймаут` — время ожидания отклика пользователя в секундах (необязательный параметр).

3. `ВыбратьКаталог()`.

Метод `ВыбратьКаталог(ИмяКаталога,ЗаголовокОкна,Таймаут)` открывает окно диалога выбора каталога.

Возвращает:

- 0 — если в окне диалога нажата кнопка **Отмена**;
- 1 — если в окне диалога нажата кнопка **ОК**, при этом в переменную `ИмяКаталога` возвращается имя выбранного каталога;
- −1 (минус единица) — закончилось время (Таймаут) ожидания отклика пользователя.

Параметры:

- `ИмяКаталога` — переменная, содержащая на входе строку с именем начального каталога, на выходе имя выбранного каталога;
- `ЗаголовокОкна` — строка с заголовком окна;
- `Таймаут` — время ожидания отклика пользователя в секундах (необязательный параметр).

4. `СуществуетФайл()`.

Метод `СуществуетФайл(ИмяФайла)` проверяет существование файла.

Возвращает:

- 1 — файл существует;
- 0 — не существует.

Параметр — `ИмяФайла` определяет строковое выражение с именем файла.

5. `КопироватьФайл()`.

Метод `КопироватьФайл(ИмяФайлаИсточника,ИмяФайлаПриемника,ФлагПерезаписи)` копирует файл.

Параметры:

- `ИмяФайлаИсточника` — строка с именем файла-источника;
- `ИмяФайлаПриемника` — строка с именем файла-приемника;
- `ФлагПерезаписи`:
  - 0 — существующий файл приемника перезаписать;
  - 1 — существующий файл приемника не перезаписывать.

6. `УдалитьФайл()`.

Метод `УдалитьФайл(ИмяФайла)` удаляет файл.

Параметр — `ИмяФайла` определяет строка с именем удаляемого файла.

7. `ПереименоватьФайл()`.

Метод `ПереименоватьФайл(ИмяФайлаИсточника,ИмяФайлаПриемника, ФлагПерезаписи)` позволяет переименовать или переместить файл.

Параметры:

- `ИмяФайлаИсточника` — строка с именем файла-источника;

- `ИмяФайлаПриемника` — строка с новым именем файла;
- `ФлагПерезаписи`:
  - 0 — запрещает перемещение файла между дисками и существующий файл приемника не перезаписывается;
  - 1 — разрешает перемещение файла между дисками (только для файлов) и существующий файл приемника перезаписывается.

8. `НайтиПервыйФайл()`.

Метод `НайтиПервыйФайл(МаскаИмени)` открывает выборку файлов по заданной маске и находит первый файл. Метод возвращает строку с именем найденного файла.

Параметры — `МаскаИмени` определяется строкой с маской имен файлов.

В функции `ПустойКаталог()` используется данный метод во фрагменте:

```
ОФС = СоздатьОбъект("ФС");
ИФ = ОФС.НайтиПервыйФайл(ПромКаталог + "*.*");
```

9. `НайтиСледующийФайл()`.

Метод `НайтиСледующийФайл()` находит следующий файл по открытой выборке файлов. Метод возвращает строку с именем найденного файла.

В функции `ПустойКаталог()` используется данный метод в строке:

```
ИФ = ОФС.НайтиСледующийФайл();
```

10. `АтрибутыФайла()`.

Метод `АтрибутыФайла(ИмяФайла,РазмерФайла,АтрибутыФайла,ВремяСоздания, ВремяПоследнегоДоступа,ВремяПоследнейЗаписи,РасширенноеИмяФайла)` возвращает атрибуты файла (в параметрах).

Параметры:

- `ИмяФайла` — строка с именем файла;
- `РазмерФайла` — переменная, принимающая размер файла в байтах;
- `АтрибутыФайла` — переменная, принимающая атрибуты файла (возвращаемое строковое значение длиной 9 символов, в котором закодированы атрибуты файла, при этом символы могут принимать значения 0 или 1):
  - если первый символ 1 — файл только для чтения;
  - если второй символ 1 — скрытый файл;
  - если третий символ 1 — системный файл;
  - если четвертый символ 1 — каталог;
  - если пятый символ 1 — архивный файл;
  - если шестой символ 1 — обычный файл (все другие атрибуты не установлены);

- если седьмой символ 1 — временный файл;
- если восьмой символ 1 — файл, сжатый каким-либо архиватором;
- если девятый символ 1 — нет доступа к файлу;

❒ `ВремяСоздания` — переменная, принимающая строку с датой и временем создания файла;

❒ `ВремяПоследнегоДоступа` — переменная, принимающая строку с датой и временем последнего доступа к файлу;

❒ `ВремяПоследнейЗаписи` — переменная, принимающая строку с датой и временем последней записи файла;

❒ `РасширенноеИмяФайла` — переменная, принимающая строку с полным именем файла.

11. `СоздатьКаталог()`.

Метод `СоздатьКаталог(ИмяФайла)` позволяет создать новый каталог файлов.

Параметр — `ИмяФайла` представляет строку с именем создаваемого каталога.

12. `УдалитьКаталог()`.

Метод `УдалитьКаталог(ИмяФайла)` удаляет каталог файлов.

Параметр — `ИмяФайла` представляет строку с именем удаляемого каталога файлов.

13. `УстТекКаталог()`.

Метод `УстТекКаталог(ИмяФайла)` устанавливает текущий каталог файлов.

Параметр — `ИмяФайла` представляет строку с именем текущего каталога файлов.

14. `ТекКаталог()`.

Метод `ТекКаталог()` возвращает строку с именем текущего каталога файлов.

15. `WindowsКаталог()`.

Метод `WindowsКаталог()` возвращает строку с именем Windows-директории.

16. `СвободноеМестоНаДиске()`.

Метод `СвободноеМестоНаДиске(ИмяДиска)` возвращает размер свободного дискового пространства в байтах.

Параметр — `ИмяДиска` определяется строкой с именем диска (например, C:).

Методы объекта типа "XBase"

При использовании методов, предназначенных для работы с файлами баз данных формата DBF, следует обратить особое внимание на то обстоятельство,

что используется объект типа "XBase", поэтому необходимо создавать указатель на него с помощью метода `СоздатьОбъект()`. Например:

```
БД=СоздатьОбъект("XBase");
```

Необходимо также иметь в виду, что при использовании методов объекта "Xbase" имена полей в файлах баз данных формата DBF должны состоять только из английского алфавита на верхнем регистре. Рассмотрим методы этой группы.

1. `СоздатьФайл()`.

Метод `СоздатьФайл(ПутьКБазе,ПутьКИндексу)` позволяет создать новый файл DBF-формата.

Параметры:

- `ПутьКБазе` — строковое выражение, определяющее путь доступа к файлу базы DBF-формата;
- `ПутьКИндексу` — строковое выражение, определяющее путь доступа к индексному файлу.

2. `ОткрытьФайл()`.

Метод `ОткрытьФайл(ПутьКБазе,ПутьКИндексу,ТолькоЧтение)` позволяет открыть существующую базу.

Параметры:

- `ПутьКБазе` — строковое выражение, определяющее путь доступа к файлу базы DBF-формата;
- `ПутьКИндексу` — строковое выражение, определяющее путь доступа к индексному файлу базы;
- `ТолькоЧтение` — число (необязательный параметр), принимающее значение:
  - 1 — файл открывается в режиме только чтение;
  - 0 — файл открывается в режиме полного доступа (при этом файл открывается в эксклюзивном режиме).

В процедуре `Выгрузить()` данный метод используется в следующем фрагменте:

```
ФС.ВыбратьФайл(0,ИмяФайла,Каталог,"Выберите файл выгрузки
               """+Наименование+""""",
               "Файлы прописи(*.dbf)|*.dbf");
БД.ОткрытьФайл(Каталог+ИмяФайла,"",0);
```

3. `Открыта()`.

Метод `Открыта()` возвращает значение флага открытия файла:

- 1 — база открыта;
- 0 — не открыта.

4. ЗакрытьФайл().

Метод ЗакрытьФайл() позволяет закрыть ранее открытую или созданную базу.

В процедуре Выгрузить() данный метод используется в строке:

```
БД.ЗакрытьФайл();
```

5. ОчиститьФайл().

Метод ОчиститьФайл() позволяет очистить все записи в базе.

В процедуре Выгрузить() данный метод используется в строке:

```
БД.ОчиститьФайл();
```

6. Сжать().

Метод Сжать() позволяет сжать базу, убрав удаленные записи.

7. Переиндексировать().

Метод Переиндексировать() позволяет переиндексировать базу.

8. ПоказыватьУдаленные().

Метод ПоказыватьУдаленные(Режим) позволяет установить режим показа удаленных записей в базе. Возвращает текущее числовое значение режима показа удаленных записей в базе (на момент до исполнения метода).

Параметр:

- Режим — представляет число, принимающее значение:
  - 1 — показывать удаленные записи,
  - 0 — нет (по умолчанию устанавливается — 0).

9. Первая().

Метод Первая() позволяет перейти на первую запись.

Возвращает:

- 1 — если действие выполнено;
- 0 — если действие не выполнено.

10. Последняя().

Метод Последняя() позволяет перейти на последнюю запись.

Возвращает:

- 1 — если действие выполнено;
- 0 — если действие не выполнено.

11. Следующая().

Метод Следующая() позволяет перейти на следующую запись.

Возвращает:

- 1 — если получена следующая запись;
- 0 — если не получена следующая запись.

12. `Предыдущая()`.

Метод `Предыдущая()` позволяет перейти на предыдущую запись.

Возвращает:

- 1 — если получена предыдущая запись;
- 0 — если не получена предыдущая запись.

13. `НомерЗаписи()`.

Метод `НомерЗаписи()` возвращает внутренний номер текущей записи.

14. `Перейти()`.

Метод `Перейти(НомерЗаписи)` позволяет перейти на запись по ее номеру записи.

Параметр — `НомерЗаписи` определяет внутренний номер записи в базе данных.

15. `ВКонце()`.

Метод `ВКонце()` возвращает значение флага конца файла:

- 1 — конец файла,
- 0 — нет.

16. `ВНачале()`.

Метод `ВНачале()` возвращает значение флага начала файла:

- 1 — начало файла,
- 0 — нет.

17. `ТекущийИндекс()`.

Метод `ТекущийИндекс(НазваниеИндекса)` возвращает строку с названием текущего индекса (на момент до выполнения метода).

Параметр — `НазваниеИндекса` определяет выражение с названием индекса.

18. `Найти()`.

Метод `Найти(Ключ,Режим)` позволяет найти запись по индексу.

Возвращает:

- 1 — если действие выполнено (запись найдена);
- 0 — если действие не выполнено.

Параметры:

- `Ключ` — выражение со значением ключа текущего индекса;

- Режим — режим поиска записей:
  - 0 — ищет запись на точное соответствие ключу (сравнение типа =);
  - 1 — ищет запись на точное соответствие с ключом или большую (сравнение типа >=);
  - 2 — ищет запись с большим ключом (сравнение типа >);
  - −1 (минус единица) — ищет запись на точное соответствие с ключом или меньшую (сравнение типа <=);
  - −2 (минус два) — ищет запись с меньшим ключом (сравнение типа <).

19. НайтиПоКлючу().

Метод НайтиПоКлючу(Режим) позволяет найти запись по индексу. Перед вызовом метода следует установить значения всех атрибутов агрегатного объекта типа "Ключ", которые участвуют в вычислении выражения текущего индекса.

Возвращает:

- 1 — если действие выполнено (запись найдена);
- 0 — если действие не выполнено.

Параметры:

Режим — число, определяющее режим поиска записей и принимающее значение:

- 0 — ищет запись на точное соответствие ключу (соответствие типа =);
- 1 — ищет запись на точное соответствие с ключом или большую (соответствие типа >=);
- 2 — ищет запись с большим ключом (соответствие типа >);
- −1 (минус единица) — ищет запись на точное соответствие с ключом или меньшую (соответствие типа <=);
- −2 (минус два) — ищет запись с меньшим ключом (соответствие типа <).

20. ПолучитьЗначениеПоля().

Метод ПолучитьЗначениеПоля(НазваниеПоля) возвращает значение поля.

Параметр — НазваниеПоля определяет выражение с названием поля или с номером поля.

21. УстановитьЗначениеПоля().

Метод УстановитьЗначениеПоля(НазваниеПоля,Значение) позволяет установить значение поля текущей записи.

Параметры:

- НазваниеПоля — выражение с названием поля или с номером поля;
- Значение — значение поля.

22. `Добавить()`.

Метод `Добавить()` позволяет добавить новую пустую запись.

В процедуре `Выгрузить()` данный метод используется в строке:

```
БД.Добавить();
```

23. `Скопировать()`.

Метод `Скопировать()` позволяет скопировать текущую запись.

24. `АвтоСохранение()`.

Метод `АвтоСохранение(Режим)` позволяет установить режим автоматического сохранения изменений в базе. Метод возвращает текущее числовое значение режима автоматического сохранения изменений в базе (на момент до исполнения метода).

Параметры:

`Режим` — число, принимающее значения:

- 1 — задается режим автоматического сохранения изменений в базе;
- 0 — снимается автоматическое сохранение изменений в базе (по умолчанию — 0).

25. `Записать()`.

Метод `Записать()` позволяет записать изменения в базу.

26. `Отменить()`.

Метод `Отменить()` позволяет отменить запись изменения в базу.

27. `Удалить()`.

Метод `Удалить()` позволяет удалить текущую запись.

28. `ЗаписьУдалена()`.

Метод `ЗаписьУдалена()` возвращает значение флага пометки удаления текущей записи в виде числа:

- 1 — запись помечена на удаление,
- 0 — нет.

29. `Восстановить()`.

Метод `Восстановить()` позволяет восстановить текущую запись.

30. `Очистить()`.

Метод `Очистить()` позволяет очистить текущую запись.

31. `КоличествоЗаписей()`.

Метод `КоличествоЗаписей()` возвращает количество записей в базе.

32. `КоличествоПолей()`.

Метод `КоличествоПолей()` возвращает количество полей в записи.

33. КоличествоИндексов().

Метод КоличествоИндексов() возвращает количество индексов в базе.

34. ОписаниеПоля().

Метод ОписаниеПоля(НомерПоля,НазваниеПоля,Тип,Длина,Точность) возвращает в параметрах описание поля с заданным номером.

Параметры:

- НомерПоля — номер поля для которого требуется получить описание;
- НазваниеПоля — идентификатор переменной для названия поля;
- Тип — идентификатор переменной для типа поля;
- Длина — идентификатор переменной для длины поля;
- Точность — идентификатор переменной для числа знаков после десятичной точки (имеет смысл только для числовых полей).

35. ОписаниеИндекса().

Метод ОписаниеИндекса(НомерИндекса,НазваниеИндекса,Выражение,Уникальность,Убывание,Фильтр) возвращает в параметрах описание индекса с заданным номером.

Параметры:

- НомерИндекса — номер индекса, для которого требуется получить описание;
- НазваниеИндекса — идентификатор переменной названия индекса;
- Выражение — идентификатор переменной ключа индекса в соответствии с инструкциями CodeBase;
- Уникальность — идентификатор переменной для флага уникальности индекса:
  - 1 — уникальный;
  - 0 — нет;
- Убывание — идентификатор переменной для флага направления убывания индекса:
  - 1 — по убыванию значения ключа;
  - 0 — по возрастанию ключа;
- Фильтр — идентификатор переменной фильтра индекса в соответствии с инструкциями CodeBase.

36. НомерПоля().

Метод НомерПоля(НазваниеПоля) возвращает номер поля.

Параметр — НазваниеПоля определяет выражение с названием поля.

37. `ДобавитьПоле()`.

Метод `ДобавитьПоле(Название,Тип,Длина,Точность)` позволяет добавить поле в структуру базы. Метод можно использовать только перед созданием новой базы.

Параметры:

- `Название` — строковое выражение, содержащее имя создаваемого поля;
- `Тип` — выражение, содержащее тип создаваемого поля;
- `Длина` — общая длина создаваемого поля;
- `Точность` — число знаков после десятичной точки (только для числовых полей).

38. `ДобавитьИндекс()`.

Метод `ДобавитьИндекс(Название,Выражение,Уникальность,Убывание,Фильтр)` позволяет добавить индекс в структуру базы.

Метод можно использовать только перед созданием новой базы.

Параметры:

- `Название` — строковое выражение, содержащее имя создаваемого индекса;
- `Выражение` — строковое выражение, содержащее ключ индекса (должно соответствовать инструкциям CodeBase);
- `Уникальность` — флаг уникальности индекса:
  - 1 — флаг уникальный;
  - 0 — нет;
- `Убывание` — флаг направления убывания индекса:
  - 1 — по убыванию значения ключа;
  - 0 — по возрастанию значения ключа;
- `Фильтр` — строковое выражение, содержащее фильтр индекса (в соответствии с инструкциями CodeBase).

39. `СоздатьИндексныйФайл()`.

Метод `СоздатьИндексныйФайл(ИмяФайла)` создает индексный файл, содержащий все индексы, которые были созданы методом `ДобавитьИндекс()`.

Параметр — `ИмяФайла` определяет строковое выражение с именем создаваемого индексного файла.

40. `КодоваяСтраница()`.

Метод `КодоваяСтраница(Режим)` позволяет установить режим кодировки для чтения и записи строковых значений в файл. Метод возвращает текущее числовое значение режима кодировки (на момент до исполнения метода).

Параметры:

- Режим:
  - 0 — Windows-кодировка;
  - 1 — DOS-кодировка.

В процедуре `Выгрузить()` данный метод используется в строке:

```
БД.КодоваяСтраница(1);
```

41. `КодОшибки()`.

Метод `КодОшибки()` возвращает код последней ошибки.

Методы работы с текстовыми файлами

Прежде чем пользоваться методами работы с текстовым файлом, программист должен четко определить, какого рода и объема информация формируется в файле. Есть два варианта формирования информации в текстовый файле:

1. Формируется текст в памяти и затем построчно или поблочно записывается в файл. Естественно, объем информации должен быть сравнительно небольшим. В данном варианте проще пользоваться методами объекта типа "Текст", которые предоставляют больше возможностей по форматированию текста.
2. Формируются строки и записываются в файл с символом "Конец строки". При записи необходимо иметь в виду, что запись последовательная и возврата к записанной строке нет. Этот способ наиболее удобен для последовательного чтения данных текстового файла.

Методы объекта типа "Текст"

При использовании методов первого варианта следует обратить особое внимание на то обстоятельство, что используется объект типа "Текст", поэтому необходимо создавать указатель на него с помощью метода `СоздатьОбъект()`:

```
ТекстДляПоказа = СоздатьОбъект("Текст");
```

Рассмотрим процедуру `ТекстФайла()`, которая открывает текстовый файл обмена данными нажатием кнопки **Просмотр**. Параметр Заголовок задает заголовок окна просмотра и редактирования файла с именем, заданным параметром ИмяФайла. Рассмотрим также процедуру `ВыгрузитьТекст()`, которая формирует файл обмена данными, для экспорта платежных документов в систему типа "Клиент-банк".

```
Функция ТекстФайла(ИмяФайла, Заголовок = "")
  ЗагрРасчетныеСчета.УдалитьВсе(); ЗагрВидыДокументов.УдалитьВсе();
  Если ФС.СуществуетФайл(ИмяФайла)=0 Тогда
```

```
    Предупреждение("Указанный файл не существует!", 5);
    Возврат(ПолучитьПустоеЗначение());
  КонецЕсли;
  Текст=СоздатьОбъект("Текст");
  Текст.КодоваяСтраница(Кодировка.ПолучитьЗначение
                        ( Кодировка.ТекущаяСтрока()));
  Текст.Открыть(ИмяФайла);
  Если Текст.КоличествоСтрок()<1 Тогда
    Предупреждение("Указанный файл не является файлом обмена!", 5);
    Возврат(ПолучитьПустоеЗначение());
  ИначеЕсли СокрЛП(Текст.ПолучитьСтроку(1))<>"1CClientBankExchange" Тогда
    Предупреждение("Указанный файл не является файлом обмена!", 5);
    Возврат(ПолучитьПустоеЗначение());
  КонецЕсли;
  Если ПустаяСтрока(Заголовок) = 0 Тогда
    Текст.Показать(Заголовок + ":", ИмяФайла);
  КонецЕсли;
  Возврат(Текст);
КонецФункции // ТекстФайла()
```

Рассмотрим процедуру `ВыгрузитьТекст()`, которая формирует файл обмена данными, для экспорта платежных документов в систему типа "Клиент банка".

```
Процедура ВыгрузитьТекст()
  Перем Вид, Часы, Минуты, Секунды, Кодир, Тип, Заголовок;
  Если КаталогИмяФайла(ИмяФайлаВыгрузки, "", "") = 0 Тогда
    Возврат;
  КонецЕсли;
  Текст=СоздатьОбъект("Текст");
  Текст.КодоваяСтраница(Кодировка.ПолучитьЗначение( Кодировка.
                        ТекущаяСтрока(), Кодир));
  ТекущееВремя(Часы, Минуты, Секунды);
    Время=Шаблон("[Часы#Ч(0)2]:[Минуты#Ч(0)2]:[Секунды#Ч(0)2]");
  Текст.ДобавитьСтроку("1CClientBankExchange");
  Текст.ДобавитьСтроку("ВерсияФормата=1.01");
  Текст.ДобавитьСтроку("Кодировка="+Кодир);
  Текст.ДобавитьСтроку("Отправитель="+Метаданные.Представление());
  Текст.ДобавитьСтроку("Получатель="+КлиентБанка);
  Текст.ДобавитьСтроку("ДатаСоздания="+
                       Формат(ТекущаяДата(), "ДДДММГГГГ"));
  Текст.ДобавитьСтроку("ВремяСоздания="+Время);
```

```
  Текст.ДобавитьСтроку("ДатаНачала="+Формат(НачДатаВыгрузки,
"ДДДММГГГГ"));
  Текст.ДобавитьСтроку("ДатаКонца="+
                       Формат(КонДатаВыгрузки, "ДДДММГГГГ"));
  Для К=1 по ВыгрРасчетныеСчета.РазмерСписка() Цикл
     Если ВыгрРасчетныеСчета.Пометка(К)=1 Тогда
        Счет=ВыгрРасчетныеСчета.ПолучитьЗначение(К);
        Текст.ДобавитьСтроку("РасчСчет="+Счет.Номер);
     КонецЕсли;
  КонецЦикла;
  Для К=1 по ВыгрВидыДокументов.РазмерСписка() Цикл
    Если ВыгрВидыДокументов.Пометка(К)=1 Тогда
       ВыгрВидыДокументов.ПолучитьЗначение(К, Вид);
       Текст.ДобавитьСтроку("Документ="+Вид);
    КонецЕсли;
  КонецЦикла;
  ТаблицаДокументов.УдалитьСтроки();
  Для К=1 по ВыгрВидыДокументов.РазмерСписка() Цикл
     Если ВыгрВидыДокументов.Пометка(К)=0 Тогда
        Продолжить;
     КонецЕсли;
     Объект=СоздатьОбъект("Документ."+
                    ВыгрВидыДокументов.ПолучитьЗначение(К, Вид));
     Состояние("Обработка: "+Вид);
     Объект.ВыбратьДокументы(НачДатаВыгрузки, КонДатаВыгрузки);
     Пока Объект.ПолучитьДокумент()=1 Цикл
        Если Объект.ПометкаУдаления()=1 Тогда
           Продолжить;
        КонецЕсли;
        ВыгрузитьПлатежныйДокумент(Объект);
     КонецЦикла;
  КонецЦикла;
  Для Стр=1 по ТаблицаДокументов.КоличествоСтрок() Цикл
     Для Кол=1 по ТаблицаДокументов.КоличествоКолонок() Цикл
        Значение=ТаблицаДокументов.ПолучитьЗначение(Стр, Кол);
        Имя=ТаблицаДокументов.ПолучитьПараметрыКолонки(Кол,Тип,,,
                                                  Заголовок);
```

```
        Если ПустоеЗначение(Значение)=1 Тогда
            Если ОбязательныеАтрибуты.Принадлежит(Имя)=1 Тогда
                ТаблицаДокументов.ПолучитьСтрокуПоНомеру(Стр);
                Сообщить(ТаблицаДокументов.ПолучитьЗначение(Стр,
                "СекцияДокумент")+" № " +
                ТаблицаДокументов.ПолучитьЗначение(Стр, "Номер")+
                " от "+ТаблицаДокументов.ПолучитьЗначение(Стр, "Дата")+
                ": не заполнено поле """+Заголовок+""".", "!");
            КонецЕсли;
            Продолжить;
        КонецЕсли;
        Если Тип="Число" Тогда
            Значение=Формат(Значение, "Ч.2");
        ИначеЕсли Тип="Дата" Тогда
            Значение=Формат(Значение, "ДДДММГГГГ");
        КонецЕсли;
        Текст.ДобавитьСтроку(Имя+"="+Значение);
    КонецЦикла;
    Текст.ДобавитьСтроку("КонецДокумента");
КонецЦикла;
Текст.ДобавитьСтроку("КонецФайла");
Текст.Записать(ИмяФайлаВыгрузки);
КонецПроцедуры // ВыгрузитьТекст()
```

1. `КоличествоСтрок()`.

Метод `КоличествоСтрок()` возвращает число строк в тексте. Например:

```
Если Текст.КоличествоСтрок()<1 Тогда
    Предупреждение("Указанный файл не является файлом обмена!", 5);
```

В процедуре `ТекстФайла()` данный метод используется во фрагменте:

```
Если Текст.КоличествоСтрок()<1 Тогда
Предупреждение("Указанный файл не является файлом обмена!", 5);
```

2. `ПолучитьСтроку()`.

Метод `ПолучитьСтроку(НомерСтроки)` возвращает строку текста с заданным номером.

Параметр — `НомерСтроки` определяет номер строки.

В процедуре `ТекстФайла()` данный метод используется во фрагменте:

```
ИначеЕсли СокрЛП(Текст.ПолучитьСтроку(1))<>"1CClientBankExchange" Тогда
    Предупреждение("Указанный файл не является файлом обмена!", 5);
```

3. `Открыть()`.

Метод `Открыть(ИмяФайла)` позволяет открыть текстовый файл.

Параметр — `ИмяФайла` определяет строку с именем файла.

В процедуре `ТекстФайла()` данный метод используется во фрагменте:

```
Текст.Открыть(ИмяФайла);
```

4. `Шаблон()`.

Метод `Шаблон(Флаг)` позволяет установить флаг добавления строк по шаблону.

Метод возвращает текущее числовое значение флага добавления строк по шаблону (на момент до исполнения метода).

Параметры:

- `Флаг` — число, принимающее значение:
  - 1 — установить флаг добавления строк по шаблону;
  - 0 — отменить.

5. `ФиксШаблон()`.

Метод `ФиксШаблон(Флаг)` позволяет установить флаг добавления строк по фиксированному шаблону.

Метод возвращает текущее числовое значение флага добавления строк по фиксированному шаблону (на момент до исполнения метода).

Параметры:

- `Флаг` — число, принимающее значение:
  - 1 — установить флаг добавления строк по фиксированному шаблону;
  - 0 — отменить установку.

6. `ВставитьСтроку()`.

Метод `ВставитьСтроку(НомерСтроки,Строка)` позволяет вставить строку с указанным номером.

Параметры:

- `НомерСтроки` — номер вставляемой строки;
- `Строка` — вставляемая строка.

7. `ДобавитьСтроку()`.

Метод `ДобавитьСтроку(Строка)` позволяет добавить строку в конец текста.

Параметр — `Строка` определяет строку текста, которая добавляется.

В процедуре `ВыгрузитьТекст()` используется данный метод в строке:

```
Текст.ДобавитьСтроку("1CClientBankExchange");
```

8. `ЗаменитьСтроку()`.

Метод `ЗаменитьСтроку(НомерСтроки,Строка)` позволяет заменить строку с указанным номером.

Параметры:

- `НомерСтроки` — номер заменяемой строки;
- `Строка` — заменяющая строка.

9. `УдалитьСтроку()`.

Метод `УдалитьСтроку(НомерСтроки)` позволяет удалить строку с указанным номером.

Параметр — `НомерСтроки` определяет номер удаляемой строки.

10. `ТолькоПросмотр()`.

Метод `ТолькоПросмотр(Флаг)` позволяет установить флаг возможности редактирования текста.

Метод возвращает текущее числовое значение режима редактирования текста (на момент до исполнения метода).

Параметры:

- `Флаг` — число, принимающее значение:
  - 1 — только просмотр текста;
  - 0 — допускается редактировать текст.

11. `Показать()`.

Метод `Показать(Заголовок,ИмяФайла)` позволяет открыть окно редактирования текста.

Параметры:

- `Заголовок` — заголовок окна редактирования;
- `ИмяФайла` — строка с именем файла.

В процедуре `ТекстФайла()` данный метод используется в строке:

```
Текст.Показать(Заголовок + ":", ИмяФайла);
```

12. `Очистить()`.

Метод `Очистить()` очищает содержимое текстового документа. Его использование позволяет заново заполнить содержимое текстового документа.

13. `КодоваяСтраница()`.

Метод `КодоваяСтраница(Режим)` позволяет установить режим кодировки текста.

Метод возвращает текущее числовое значение режима кодировки (на момент до исполнения метода).

Параметры:

- Режим:
  - 0 — Windows-кодировка;
  - 1 — DOS-кодировка.

В процедуре `ВыгрузитьТекст()` данный метод используется в следующем фрагменте кода:

```
Текст.КодоваяСтраница(Кодировка.ПолучитьЗначение
                      (Кодировка.ТекущаяСтрока(), Кодир));
```

14. `Записать()`.

Метод `Записать(ИмяФайла)` позволяет записать текст в файл с указанным именем.

Параметр — `ИмяФайла` определяет строку с именем файла.

В процедуре `ВыгрузитьТекст()`данный метод используется в следующей строке кода:

```
Текст.Записать(ИмяФайлаВыгрузки);
```

Методы чтения и записи текстовых файлов

При использовании методов, реализующих второй вариант обработки текстовых файлов, особое внимание следует обратить на то обстоятельство, что здесь используется объект типа "V7TextFile". Поэтому необходимо создавать указатель на этот объект с помощью метода `СоздатьОбъект()`. Например:

```
ТекстДляПоказа = СоздатьОбъект("V7TextFile");
```

Рассмотрим более подробно методы данной группы.

1. `ОткрытьФайл()`.

Метод `ОткрытьФайл(Имя,Режим)` открывает текстовый файл.

Параметры:

- `Имя` — имя файла (строковое выражение);
- `Режим` — число, которое может принимать значение:
  - 0 — только для чтения (после открытия допускается только чтение);
  - 1 — открывать на запись эксклюзивно (при открытии, если файла нет, — он создается, если есть — очищается).

По умолчанию устанавливается значение — 0.

2. `ЗакрытьФайл()`.

Метод `ЗакрытьФайл()` закрывает текстовый файл.

3. `ЗаписатьСтроку()`.

Метод `ЗаписатьСтроку(Строка)` записывает строку в конец файла.

Параметр — `Строка` определяет строковое выражение.

Внимание

Метод добавляет разделитель строки в конце, если в строке есть разделители, то они тоже записываются.

4. `ПрочитатьСтроку()`.

Метод `ПрочитатьСтроку(Строка)` считывает очередную строку и возвращает ее в параметр `Строка` без разделителя строк.

Возвращаемое значение:

- 1 — строка считана;
- 0 — больше строк нет.

Внимание

Если в файле есть символ — `EndOfFile`, метод `ПрочитатьСтроку()` обрабатывает его так же, как объект типа "Текст".

5. `ВыбратьСтроки()`.

Метод `ВыбратьСтроки()` инициирует считывание с начала файла.

6. `РазмерБуфера()`.

Метод `РазмерБуфера(Размер)` устанавливает размер внутреннего буфера.

Параметр — `Размер` устанавливает размер внутреннего буфера.

Если параметр не задан, метод возвращает текущее значение размера внутреннего буфера.

7. `КодоваяСтраница()`.

Метод `КодоваяСтраница(Страница)` устанавливает кодовую страницу.

Параметры:

- **Страница** — число, принимающее значение:
  - 0 — Windows-кодировка;
  - 1 — DOS-кодировка.

Если параметр не задан, метод возвращает текущее значение кодовой страницы. Если параметр задан, то метод возвращает прежнее значение кодовой страницы.

Глава 5

Работа над ошибками

Работа над ошибками при разработке процедур разделена на два этапа:

1. Обнаружение синтаксических ошибок в тексте модуля и тексте запроса. Реализуется в среде конфигуратора.
2. Анализ и исправление ошибок, определенных системой в процессе выполнения модуля. К таким ошибкам, например, можно отнести: ошибки в задании атрибутов или методов; неправильное определение количественного или качественного состава параметров в методе и др.

Синтаксический контроль

Предварительный синтаксический контроль проводится в режиме "Синтаксический контроль". Этот режим, как показано на рис. 5.1, инициализируется из меню Конфигуратора — **Действия | Синтаксический контроль** (для проведения синтаксического контроля в тексте модуля) и **Действия | Синтаксический контроль запросов** (для контроля в тексте запросов).

Нарушение синтаксических конструкций

Самой простой и распространенной ошибкой синтаксиса является пропуск символа точка с запятой (;), завершающего некоторый оператор. Сообщение о данной ошибке предельно ясно указывает на ее характер и строку, в которой она была допущена (см. рис. 5.2)

При нарушении синтаксических конструкций `Если…тогда…КонецЕсли` выдается сообщение об ошибке типа

Ожидается ключевое слово.

Следует обратить особое внимание на то обстоятельство, что в этом случае сообщается номер той строки модуля, в которой система обнаружила логическое несоответствие. Этот номер необязательно ссылается на ошибочную конструкцию. Ссылка делается на последний оператор, который не был

закрыт операторной скобкой. Соответственно, и ключевое слово в этом случае будет указываться со смещением (рис. 5.3.). Сообщения о подобных ошибках завершаются строкой:

Обнаружено логическое завершение исходного текста модуля.

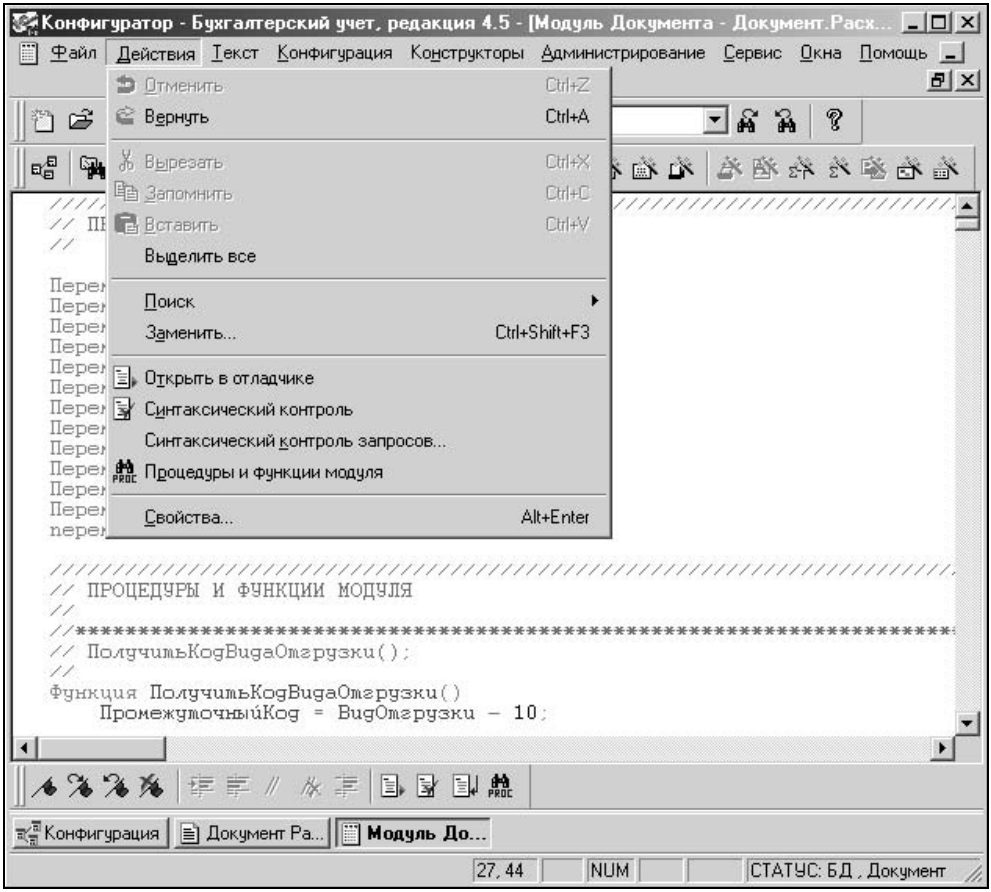


Рис. 5.1. Меню Конфигуратора для запуска синтаксического контроля

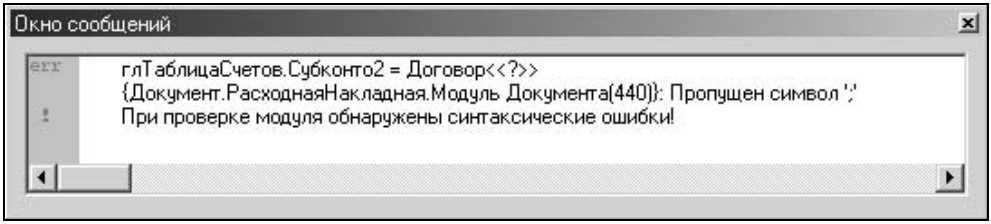


Рис. 5.2. Окно сообщений при обнаружении пропуска символа точка с запятой

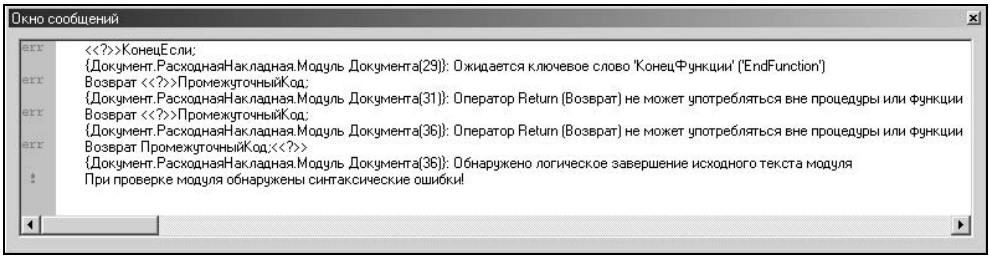


Рис. 5.3. Окно сообщений при обнаружении пропуска ключевого слова `Если`

Рассмотрим действия, которые надо выполнить при обнаружении сообщения об ошибке.

Первое, что надо сделать, — это внимательно проверить все вложенные конструкции.

В приведенном на рис. 5.3 примере, в исходном тексте модуля было пропущено ключевое слово `Если`. Теперь рассмотрим сообщения об ошибке в исходном тексте модуля, когда было пропущено ключевое слово `Тогда` (рис. 5.4).

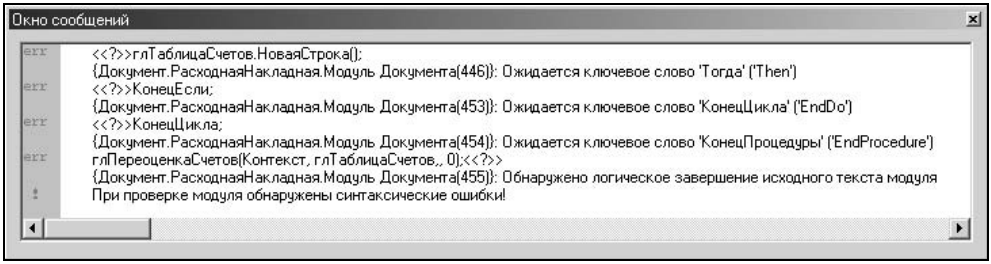


Рис. 5.4. Окно сообщений при обнаружении пропуска ключевого слова `Тогда`

В данном случае сообщение более конкретизировано, т. к. основное ключевое слово `Если` все же есть в тексте программы. Здесь следует обратить внимание на сгенерированные далее сообщения. Они являются следствием основной ошибки, после исправления которой генерироваться не будут.

При нарушении синтаксических конструкций `Пока...Цикл...КонецЦикла` выдается сообщение об ошибке типа

Ожидается ключевое слово.

Особое внимание следует обратить на то обстоятельство, что в сообщении говорится об отсутствии ключевого слова той конструкции, которая была нарушена лишним закрывающим элементом. Соответственно указывается и ключевое слово нарушенного уровня (рис. 5.5.). Сообщения о подобных ошибках завершаются строкой:

Обнаружено логическое завершение исходного текста модуля.

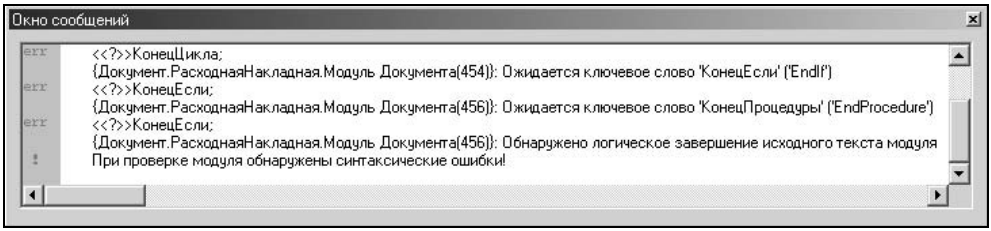


Рис. 5.5. Окно сообщений при обнаружении пропуска ключевого слова `Пока` или `Для`

В приведенном на рисунке примере, в исходном тексте модуля, было пропущено ключевое слово `Пока`. Рассмотрим сообщения об ошибке в исходном тексте модуля, когда было пропущено ключевое слово `КонецЦикла` (рис. 5.6).

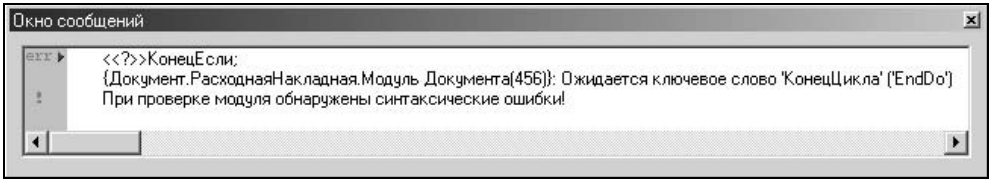


Рис. 5.6. Окно сообщений при обнаружении пропуска ключевого слова `КонецЦикла`

В данном случае сообщение более конкретизировано, т. к. основное ключевое слово `Пока` все же в тексте есть.

Отсутствие инициализации переменной

Если в тексте модуля выполняется присвоение значения переменной, которая к этому моменту была не определена, синтаксическим контролем формируется сообщение (см. рис. 5.7):

Переменная не определена [имя переменной].

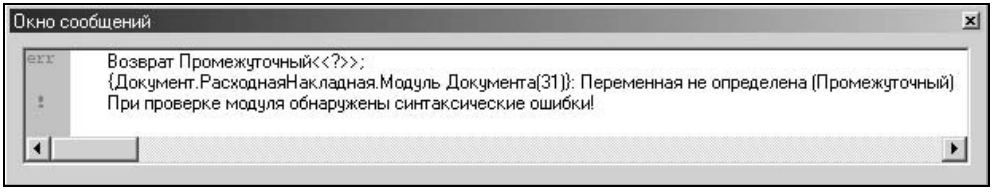


Рис. 5.7. Окно сообщений при обнаружении переменной, которая не была определена в модуле

Синтаксический контроль запросов

Синтаксический контроль не формирует сообщение об ошибке в тексте запроса. Сообщение об ошибке в тексте модуля будет сформировано при выполнении метода запроса `Выполнить()`, без конкретизации и со ссылкой на оператор, вызвавший выполнение запроса. Для исключения таких ошибок полезно проводить синтаксический контроль запросов. Предварительный синтаксический контроль запросов, как ранее указывалось, проводится с помощью режима — "Синтаксический контроль запросов", который вызывается из меню **Действия | Синтаксический контроль запросов** (рис. 5.1).

Например, присутствие лишней запятой во второй строке запроса вызовет сообщение об ошибке при проведении синтаксического контроля запросов (см. рис. 5.8).

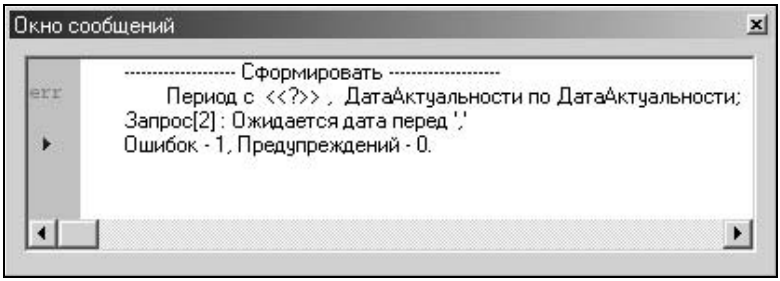


Рис. 5.8. Окно сообщений при обнаружении ошибки в строке запроса

Ошибки выполнения

Задание метода или атрибута, который не определен для данного объекта, не определяется при проведении синтаксического контроля. Ошибки при задании параметров также не относятся к ошибкам синтаксиса. Они обнаруживаются только при выполнении модуля. При обнаружении ошибки, во время выполнения модуля, в окно сообщений выводится сообщение с номером строки, в которой эта ошибка была обнаружена.

Ошибки задания методов

Ошибка, допущенная в наименовании метода, интерпретируется как использование метода, который не определен для данного объекта метаданных. При обнаружении такого рода ошибок в окно сообщений выдается сообщение (рис. 5.9):

> **Поле агрегатного объекта не обнаружено [наименование метода или атрибута]**.

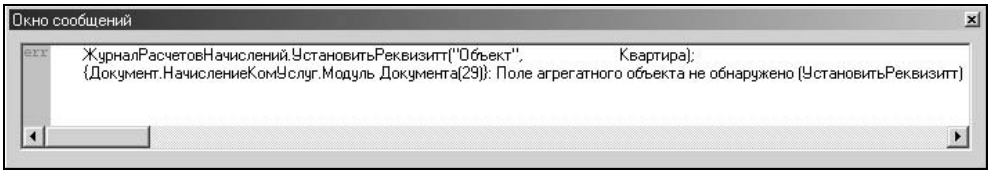


Рис. 5.9. Окно сообщений при обнаружении ошибки во время выполнения метода

Ошибки передачи параметров

Если при использовании метода было указано недостаточное количество его параметров, то в окно сообщений выводится сообщение (рис. 5.10):

> **Недостаточное количество параметров передано при вызове функции/процедуры объекта**.

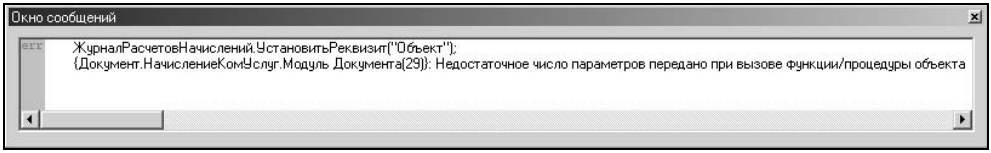


Рис. 5.10. Окно сообщений при обнаружении ошибки задания недостаточного количества параметров метода

Внимание

Нарушение порядка передачи параметров (их последовательности) не вызывает сообщения об ошибке.

При указании для метода лишнего количества параметров, в окно сообщений выводится сообщение (рис. 5.11):

> **Слишком много параметров передано при вызове функции/процедуры объекта**.

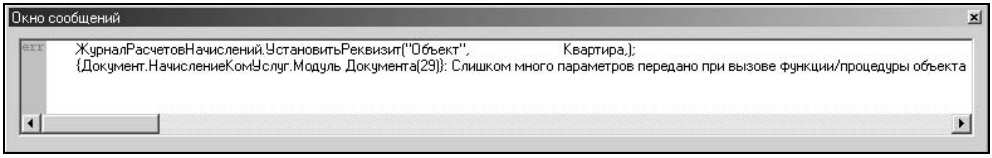


Рис. 5.11. Окно сообщений при обнаружении ошибки задания излишнего количества параметров

Ошибки специфики объекта

Для некоторых объектов метаданных, например, "Журнала расчетов", определено обязательное задание некоторых атрибутов. При записи нового объекта данного типа в окно сообщений выводится сообщение (рис. 5.12):

> **Не все обязательные реквизиты установлены при вводе новой записи журнала**.

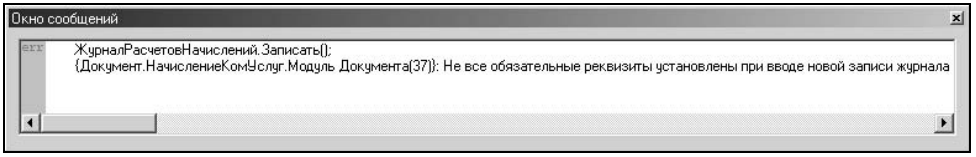


Рис. 5.12. Окно сообщений при обнаружении ошибки в задании реквизитов новой записи журнала

Для объекта метаданных типа "Операция" задание атрибутов типа "Субконто" необязательно, но вид субконто определяется заданным счетом. При обнаружении несоответствия вида субконто заданному счету в окно сообщений выдается сообщение (рис. 5.13):

> **Для счета [счет] вид субконто [Вид субконто] неопределен**.

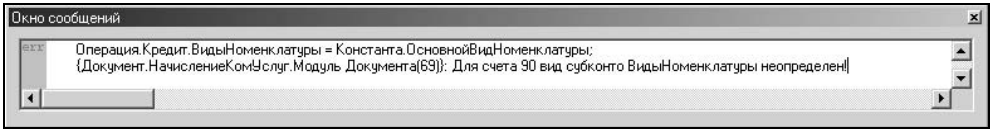


Рис. 5.13. Окно сообщений при обнаружении ошибки задания вида субконто

При нарушении соответствия типа данных в логических операциях в окно сообщений выдается сообщение:

> **Операции сравнения на больше-меньше допустимы только над значениями совпадающих базовых типов (число, строка, дата)**.

Например, при сравнении переменной типа "Дата" со строкой, будет сформировано сообщение:

> **Если ДатаДок>"15.02.2004" тогда**
>
> **{Документ.НачислениеКомУслуг.Форма.Модуль(20)}: Операции сравнения на больше-меньше допустимы только над значениями совпадающих базовых типов (число, строка, дата)**.

Внимание

Сообщение выдается только для логических операций. Для операции присвоения сообщение не генерируется.

Ошибки формирования таблиц

Для корректировки обнаруженной ошибки в тексте модуля, в окне сообщений необходимо дважды щелкнуть кнопкой мыши по номеру строки с сообщением об ошибке. В этом случае в Конфигураторе откроется окно соответствующего модуля и указатель установится на строке с ошибкой. Данный механизм отладки не предоставляется при обнаружении ошибок шаблонов таблиц. Это и понятно, т.к. формулы и выражения в ячейках типа "Выражение" и "Шаблон" невозможно соотнести со строками модуля.

Синтакс-Помощник

В затруднительных ситуациях при обнаружении ошибки полезно обратиться к "Синтакс-Помощнику". Запомнить все методы и порядок передачи параметров невозможно, получить же необходимую справку довольно просто, если использовать механизм подсказки — "Синтакс-Помощник". Программа "Синтакс-Помощника" вызывается нажатием кнопки  на панели инструментов **Конфигурация** (рис. 5.14).

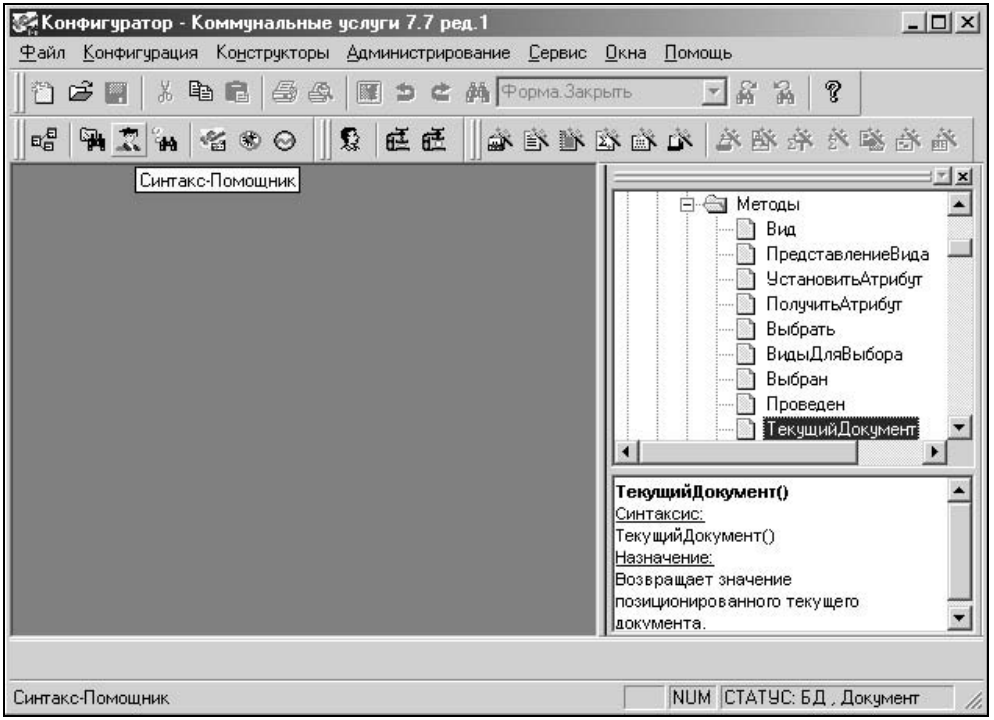


Рис. 5.14. Диалоговое окно **Конфигуратор** с открытым окном "Синтакс-Помощник"

Окно "Синтакс-Помощника" состоит их двух частей (панелей):

- ❒ Дерево тем;
- ❒ Справка.

На панели "Дерево тем", для справки, необходимо выбрать конечный пункт темы и двойным щелчком клавиши мыши открыть справку (рис. 5.15).

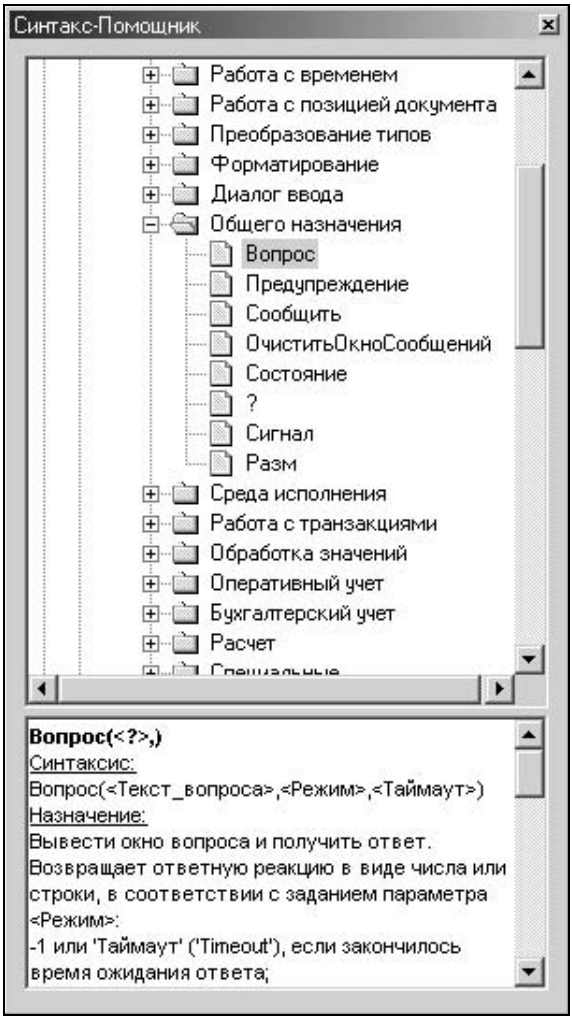


Рис. 5.15. Диалоговое окно "Синтакс-Помощника" со справкой

Сама справка при этом выводится на панели "Справка", которая расположена в нижней части окна "Синтакс-Помощника".

Глава 6

Обработка событий

Предопределенные процедуры, описанные в *гл. 2 (разд. "Предопределенные процедуры")*, предоставляют программисту механизм обработки событий, которые возникают вследствие интерактивных действий пользователя со стандартным оборудованием, т. е. клавиатурой и манипулятором типа мышь. В процессе обработки этих событий иногда требуется принятие некоторого решения от пользователя. В связи с этим, в данной главе мы рассмотрим методы формирования сообщений и ответы на них.

Для работы с нестандартным оборудованием используется предопределенная процедура `ОбработкаВнешнегоСобытия()`. Поэтому в данной главе мы также рассмотрим порядок использования данной процедуры и механизм работы со сканером штрих-кода как в относительно простой конфигурации — "Коммунальные услуги", так и более сложный вариант работы со сканером штрих-кода в конфигурации "1С:Предприятие.Торговля+Склад".

Формирование сообщений

Кроме механизма предопределенных процедур для обработки событий в системе определен ряд системных функций, с помощью которых формируются сообщения пользователю. Рассмотрим каждую из них по отдельности.

1. Функция `Вопрос()`.

Функция `Вопрос(Текст_вопроса,Режим,Таймаут)` позволяет вывести окно вопроса и получить ответ. Функция возвращает ответную реакцию на вопрос в виде числа или строки, в соответствии с заданным значением параметра `Режим`. Возможные значения, возвращаемые функцией, следующие:

- −1 или `Таймаут` (`Timeout`) — если закончилось время ожидания ответа;
- 1 или `ОК` (`OK`) — если нажата кнопка **ОК**;
- 2 или `Отмена` (`Cancel`) — если нажата кнопка **Отмена**;
- 3 или `Стоп` (`Abort`) — если нажата кнопка **Стоп**;
- 4 или `Повтор` (`Retry`) — если нажата кнопка **Повтор**;

- 5 или Пропустить (Ignore) — если нажата кнопка **Пропустить**;
- 6 или Да (Yes) — если нажата кнопка **Да**;
- 7 или Нет (No) — если нажата кнопка **Нет**.

Параметры:

- Текст_вопроса — строка текста вопроса;
- Режим — определяет набор кнопок диалога в виде числа или строки (в скобках английское написание):
  - 0 или ОК (OK) — кнопка **ОК**;
  - 1 или ОК+Отмена (OK+Cancel) — кнопки **ОК** и **Отмена**;
  - 2 или Стоп+Повтор+Пропустить (Abort+Retry+Ignore) — кнопки **Стоп**, **Повтор**, **Пропустить**;
  - 3 или Да+Нет+Отмена (Yes+No+Cancel) — кнопки **Да**, **Нет**, **Отмена**;
  - 4 или Да+Нет (Yes+No) — кнопки **Да**, **Нет**;
  - 5 или Повтор+Отмена (Retry+Cancel) — кнопки **Повтор**, **Отмена**;
  - 0 — в любом другом случае или при отсутствии параметра;
- Таймаут — число секунд времени ожидания ответа (если опущен или 0, то время ожидания без ограничения).

В модуле формы документа "Начисление ком. услуг", в процедуре, назначенной кнопке **Заполнить**, определен фрагмент с данной функцией и с заданием параметра режим в варианте 4.

```
Если КоличествоСтрок()>0 Тогда
  Если Вопрос("Перед заполнением список квартир будет очищен".
              "Продолжить?","Да+Нет")="Нет" Тогда
    Возврат
  КонецЕсли;
КонецЕсли;
```

В модуле формы журнала расчетов "Начисления" в процедуре ПриИсправленииРезультата() определен фрагмент с данной функцией и с заданием параметра режим в варианте 1.

```
Если Запись.Исправлена = 0 тогда
  Если Вопрос("Сохранить изменения результата расчета?",1,60)=2 тогда
    СтатусВозврата(0);
    Возврат;
  КонецЕсли;
КонецЕсли;
```

2. Функция `Предупреждение()`.

Функция `Предупреждение(Текст_сообщения,Таймаут)` позволяет вывести окно предупреждения (модальное). Работа в модальном окне исключает переключение на другие окна, поэтому пользователю необходимо подтвердить получение предупреждения или дождаться окончания таймаута.

Параметры:

- `Текст_сообщения` — строка текста предупреждения;
- `Таймаут` — время (в секундах) отображения окна предупреждения (если параметр опущен или равен 0, то вывод окна предупреждения по продолжительности не ограничен).

В модуле формы документа "Показания водомеров" в процедуре `ПриЗаписи()`, определен фрагмент с данной функцией:

```
Процедура ПриЗаписи()
  Если Услуга.Выбран()=0 тогда
     Сообщить("Не выбран вид расчета - услуга! Документ
     не записан");
     Предупреждение("Не выбран вид расчета - услуга! Документ
     не записан",30);
     СтатусВозврата(0);
     Возврат;
  КонецЕсли;
  Кв = СоздатьОбъект("Справочник.Квартиры");
  ВыбратьСтроки();
  Пока ПолучитьСтроку()=1 цикл
      Если Кв.НайтиПоКоду(Квартира.Код,0) = 1 тогда
         Кв.ПоказаниеХолВоды.Установить(КонМесяца(ДатаДок),
                                        ПоказаниеСчетчика);
         Кв.Записать();
      КонецЕсли;
  КонецЦикла;
КонецПроцедуры
```

3. Функция `Сообщить()`.

Функция `Сообщить(Текст_сообщения,ИмиджМаркера)` позволяет вывести строку в окно сообщений. Перед сообщениями можно отображать специальные пиктограммы, которыми можно помечать сообщения различной важности.

Параметры:

- `Текст_сообщения` — строка текста сообщения;

- ИмиджМаркера — строковое выражение, задающее тип пиктограммы, выводимой перед сообщением (необязательный параметр) и принимающее следующие значения:
  - I;
  - !;
  - !!;
  - !!!;
  - . (символ точки) — обычное сообщение;
  - символ пробела — без маркера.

Примечание

Заданная строка отображается в левом верхнем углу окна сообщений и выделяется красным цветом.

В модуле формы документа "Показания водомеров" в процедуре ПриЗаписи() определен фрагмент с данной функцией.

4. Функция ОчиститьОкноСообщений().

Функция ОчиститьОкноСообщений() позволяет очистить окно сообщений.

5. Функция Состояние().

Функция Состояние(Текст_сообщения) позволяет вывести сообщение в строку состояния.

Параметр — Текст_сообщения определяет строку с текстом сообщения.

Данная функция часто используется в цикле, при обработке строк табличной части документа. Например:

```
Состояние("Обработано: " + Окр(НомерСтроки/ОбщееКоличество*100, 0, 1) + "%");
```

6. Функция Сигнал().

Функция Сигнал() позволяет вывести звуковой сигнал.

Обработка внешних событий

Работа с нестандартным оборудованием состоит из двух этапов:

1. Настройка системы на состав оборудования, внешнюю компоненту и обработку обслуживания. Внешняя компонента поставляется поставщиком оборудования в виде библиотеки.
2. Обработка события, сгенерированного оборудованием компьютера.

Настройка производится перед началом эксплуатации оборудования и при смене модели оборудования, внешней компоненты или обработки обслуживания.

Обработка внешнего события — реакция программы на внешнее событие, генерируемое устройством, включающая в себя обмен и обработку данных, полученных с этого устройства.

Настройка оборудования

Для работы оборудования необходимо наличие драйвера, обеспечивающего взаимодействие с ним. В этом случае требуется наличие внешней компоненты, которая располагается в файле DLL. Для создания объекта управления торговым оборудованием нужен программный идентификатор внешней компоненты. Обработка обслуживания предусматривает реализацию алгоритма взаимодействия с торговым оборудованием.

Зададим обработку "Настройка оборудования", при помощи которой определяются различные параметры использования оборудования на конкретном рабочем месте (см. рис. 6.1).

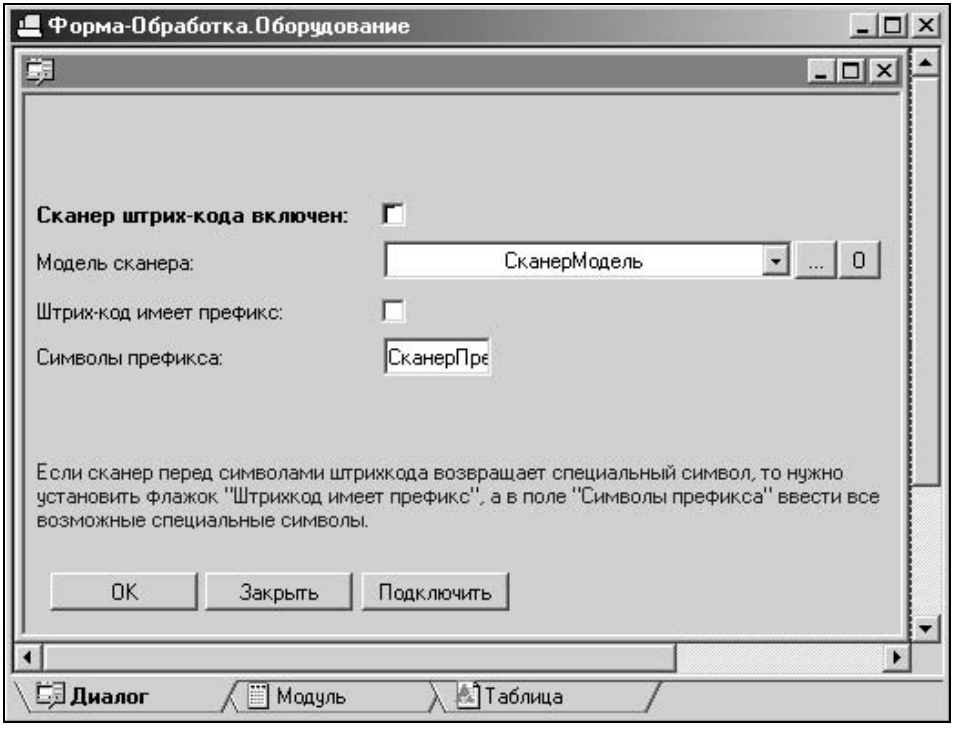


Рис. 6.1. Диалоговое окно настройки сканера

Если сканер перед символами штрих-кода возвращает специальный символ, то нужно установить флажок **Штрих-код имеет префикс**, а в поле **Символы префикса** ввести все возможные специальные символы (без запятых). Создадим обработку — "Модель" — для задания параметров модели сканера, которая вызывается при выборе реквизита **СканерМодель** (рис. 6.2).

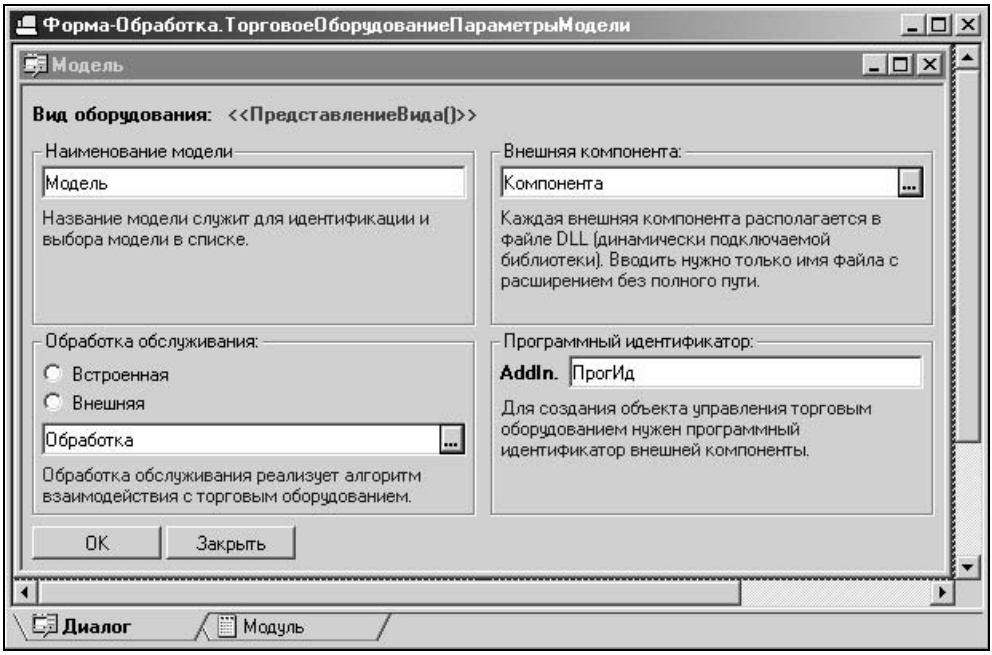


Рис. 6.2. Диалоговое окно настройки модели сканера

При выборе реквизита **СканерМодель** предлагается выбрать файл с расширением DLL. Для связи с внешней компонентой пользователем задается программный идентификатор, например, `Сканер (Scanner)`. Тип объекта метаданных задается именно таким программным идентификатором:

```
Сканер = СоздатьОбъект("AddIn." + ПрогИд);
```

После записи настройки торгового оборудования система загружает и запускает необходимые для его нормальной работы драйверы. Впоследствии, при запуске системы на данном рабочем месте, она будет, на начальном этапе, автоматически настраиваться в соответствии с записанными настройками. Кнопке **Подключить** (см. рис. 6.1) назначим процедуру `СохранитьИзменения()`, в которой определим переменные в глобальном модуле `глСканерВкл`, `глСканер`, `глСканерМодель`, `глСканерПрефикс`.

```
Процедура СохранитьИзменения()
  Перем Компонента, ПрогИд, Обработка;
```

```
// отключаем все используемое оборудование
  Если глСканерВкл = 1 Тогда
     Параметры.Установить("Объект", глСканер);
     глОборудованиеКоманда("Сканер", глСканерОбработка, Параметры);
     глСканерВкл = 0;
     глСканер    = 0;
  КонецЕсли;
  ТекСтр = СканерМодель.ТекущаяСтрока();
  Модель = "";
  Если ТекСтр > 0 Тогда
     Модель = СканерМодель.ПолучитьЗначение(ТекСтр);
  КонецЕсли;
  Если ПустоеЗначение(Модель) = 0 Тогда
     Поз = НайтиМодель(Модель);
     Если Поз = 0 Тогда
        глСканерВкл    = 0;
        глСканерМодель = "";
     Иначе
        глСканерВкл    = СканерВкл;
        глСканерМодель = Модель;
        ПолучитьПараметрМодели(Поз, "Компонента" ,
                                              глСканерКомпонента);
        ПолучитьПараметрМодели(Поз, "ПрогИд"     , глСканерПрогИд);
        ПолучитьПараметрМодели(Поз, "Обработка"  ,
                                               глСканерОбработка);
     КонецЕсли;
  КонецЕсли;
  глСканерПрефикс     = СканерПрефикс;
  глСканерЕстьПрефикс = СканерИспПрефикс;
// Подключаем оборудование
  ПодключитьОборудование();
КонецПроцедуры // СохранитьИзменения()
Функция ПолучитьПараметрМодели(Поз, Параметр, Значение)
  Значение = Модели.ПолучитьЗначение(Поз, Параметр);
  Возврат 1;
КонецФункции // ПолучитьПараметрМодели()
Процедура ПодключитьОборудование()
  Параметры = СоздатьОбъект("СписокЗначений");
// подключаем сканер
```

```
  Если глСканерВкл = 1 Тогда
// Формируем параметры для передачи в обработку обслуживания сканера
     Параметры.Установить("Процесс", "подключить");
     Параметры.Установить("Компонента", глСканерКомпонента);
     Параметры.Установить("ПрогИд"    , глСканерПрогИд);
     Если ПустоеЗначение(глСканерОбработка) = 1 Тогда
// внутренняя обработка
Если Метаданные.Обработка("ОбслуживаниеСканер").Выбран() = 1 Тогда
           ОткрытьФормуМодально("Обработка.ОбслуживаниеСканер",
                                                    Параметры);
        Иначе
           Параметры.Установить("ОписаниеРезультата",
                          "Отсутствует обработка обслуживания");
        КонецЕсли;
     Иначе
Если ФС.СуществуетФайл(глКаталогОписаний + глСканерОбработка) = 0 Тогда
           Параметры.Установить("ОписаниеРезультата",
                          "Отсутствует обработка обслуживания");
        Иначе
           ОткрытьФормуМодально("Отчет", Параметры,
                           глКаталогОписаний + глСканерОбработка);
        КонецЕсли;
     КонецЕсли;
     глСканерВкл = Параметры.Получить("Результат");
     Если глСканерВкл = 1 Тогда
        глСканер = Параметры.Получить("Объект");
     Иначе
        Сообщить("Сканер штрих-кода: ошибка при подключении", "!");
        Сообщить("    " + Параметры.Получить("ОписаниеРезультата"));
     КонецЕсли;
  КонецЕсли;
  Параметры.УдалитьВсе();
КонецПроцедуры // ПодключитьОборудование()
```

Обработка обслуживания

В тексте процедуры `ПодключитьОборудование()` присутствует вызов обработки обслуживания сканера с параметром "подключить", причем данная обработка может быть как внутренней, т. е. заданной в конфигурации, так и внешней, т. е. подгружаемой из файла. Зададим внутреннюю обработку `ОбслуживаниеСканер()` (рис. 6.3).

Рис. 6.3. Диалоговое окно обработки обслуживания сканера

Обработка загружает внешнюю компоненту и задает ссылку на объект внешней компоненты:

```
ЗагрузитьВнешнююКомпоненту(Компонента);
```

При подключении сканера создается объект с параметрами — `AddIn.`, `ПрогИд`.

```
Сканер = СоздатьОбъект("AddIn." + ПрогИд);
```

Загрузка внешней компоненты и создание ссылки на объект внешней компоненты производится после настройки оборудования каждый раз при старте системы. В предопределенной процедуре глобального модуля `ПриНачалеРаботыСистемы()` определим вызов:

```
ОткрытьФормуМодально("Обработка.Оборудование", "Подключить");
```

Перед рассмотрением процедуры обработки обслуживания сканера рассмотрим атрибуты и методы объекта типа `AddIn.`, `ПрогИд`.

Атрибуты объекта внешней компоненты

- `УстройствоВключено` — число, определяющее состояние объекта:
  - 0 — выключено;
  - 1 — включено.
- `Результат` — сообщение сканера о выполнении команды.

- ПосылкаДанных — число, определяющее режим буферизации данных:
  - 1 — компонента при получении штрих-кода сразу будет посылать его в систему;
  - 0 — компонента будет сохранять полученный штрих-код в очереди.

Методы объекта внешней компоненты

1. Подсоединить().

Метод Подсоединить("Scanner") позволяет подсоединить устройство.

Возвращает значения:

- 0 — действие выполнено успешно;
- любое значение, не равное 0 — действие выполнено неудачно.

2. Занять().

Метод Занять(Режим) позволяет получить доступ к устройству.

Параметры:

- Режим:
  - 1 — монопольный доступ;
  - 0 — разделенный доступ.

3. ОчиститьВход().

Метод ОчиститьВход() позволяет очистить буфер компоненты.

4. ОчиститьВыход().

Метод ОчиститьВыход() позволяет очистить буфер компоненты.

5. Отсоединить().

Метод Отсоединить("Scanner") позволяет отсоединить устройство.

Возвращает:

- 0 — действие выполнено успешно;
- любое значение, не равное 0 — действие выполнено неудачно.

Модуль обработки обслуживания

Особенности обработки обслуживания устройства заключается в том, что при вызове обработки не открывается окно формы обработки, но при этом выполняется функция, заданная в списке параметров "Процесс". Такой алгоритм достигается тем, что в предопределенной процедуре ПриОткрытии() задается СтатусВозврата(0).

Перечисленные в предыдущем разделе методы вынесены в одноименные функции для формирования строки возвращаемых обработкой параметров ОписаниеРезультата.

```
Перем ОписаниеРезультата;
Функция Подключить(Компонента, ПрогИд, Сканер)
//  Компонента - имя файла внешней компоненты
//  ПрогИд     - программный идентификатор
//  Сканер     - выходящий параметр - ссылка на созданный объект
// Возвращаемое значение:
//  1 - успешное завершение, 0 — произошла ошибка
// Описание:
// Загружает внешнюю компоненту из файла, имя которого задано в
// параметре
// Компонента, создает объект, программный идентификатор которого
// задан в параметре ПрогИд
  Рез = 0;
  Если ЗагрузитьВнешнююКомпоненту(Компонента) = 0 Тогда
     ОписаниеРезультата =
     | "Не удалось загрузить внешнюю компоненту """ + Компонента + """";
  Иначе
     Попытка
         Сканер = СоздатьОбъект("AddIn." + ПрогИд);
     Исключение
     КонецПопытки;
     Если ПустоеЗначение(Сканер) = 1 Тогда
        ОписаниеРезультата =
        | "Не удалось создать объект внешней компоненты с программным
        | идентификатором AddIn." + ПрогИд;
     Иначе
        Если Сканер.УстройствоВключено = 1 Тогда
           Рез = 1;
        Иначе
           Если Сканер.Подсоединить("Scanner") <> 0 Тогда
              ОписаниеРезультата =
              | "Не удалось подсоединить устройство";
           Иначе
              Если Сканер.Занять(1) <> 0 Тогда
                 ОписаниеРезультата =
                 | "Не удалось получить монопольный доступ к устройству";
              Иначе
                 Сканер.УстройствоВключено = 1;
                 Если Сканер.Результат <> 0 Тогда
                    ОписаниеРезультата =
                    | "Не удалось включить устройство";
```

```
                    Иначе
                        Рез = 1;
                    КонецЕсли;
                КонецЕсли;
            КонецЕсли;
        КонецЕсли;
    КонецЕсли;
КонецЕсли;
Если Рез = 1 Тогда
// очищаем буфер компоненты
    Сканер.ОчиститьВход();
    Сканер.ОчиститьВыход();
// включаем режим немедленной посылки данных
    Сканер.ПосылкаДанных = 1;
КонецЕсли;
Возврат Рез;
КонецФункции  // Подключить()
Функция Отключить(Сканер)
//  Сканер     - ссылка на объект внешней компоненты.
// Возвращаемое значение:   1
// Описание:
// освобождает порт, занятый сканером
  Сканер.УстройствоВключено = 0;
  Сканер.Отсоединить();
  Возврат 1;
КонецФункции // Отключить()
Функция ПосылкаДанных(Сканер, Флаг)
//  Сканер - ссылка на объект внешней компоненты.
//  Флаг  - режим буферизации
// Возвращаемое значение:
//  1 - успешное завершение, 0 - произошла ошибка
// Описание:
//  Если параметр Флаг = 1, компонента при получении штрих-кода срузу будет
//  посылать в Предприятие. Если Флаг = 0, компонента будет сохранять
//  полученные штрих-коды в очереди.
  Сканер.ПосылкаДанных = Флаг;
  Если Сканер.Результат = 0 Тогда
     Рез = 1;
```

```
   Иначе
      Рез = 0;
   КонецЕсли;
   Возврат Рез;
КонецФункции // ПосылкаДанных()
Функция ОчиститьВход(Сканер)
//  Сканер - ссылка на объект внешней компоненты.
// Возвращаемое значение:
//  1 - успешное завершение, 0 - произошла ошибка
// Описание:
//  очищает входную очередь сканера
   Сканер.ОчиститьВход();
   Сканер.ОчиститьВыход();
   Если Сканер.Результат = 0 Тогда
      Рез = 1;
   Иначе
      Рез = 0;
   КонецЕсли;
   Возврат Рез;
КонецФункции // ОчиститьВход()
Процедура ПриОткрытии()
   Перем Сканер, Вкл, Парам;
   СтатусВозврата(0);
   Форма.Параметр.Выгрузить(Парам);
   Форма.Параметр.УдалитьВсе();
   Если Парам.РазмерСписка() = 0 Тогда
      Рез = 0;
   Иначе
      Процесс = НРег(Парам.Получить("Процесс"));
      Если Процесс = "подключить" Тогда
         Компонента = Парам.Получить("Компонента");
         ПрогИд     = Парам.Получить("ПрогИд");
         Рез        = Подключить(Компонента, ПрогИд, Сканер);
         Форма.Параметр.Установить("Объект"    , Сканер);
      ИначеЕсли Процесс = "отключить" Тогда
         Рез = Отключить(Парам.Получить("Объект"));
      ИначеЕсли Процесс = "посылка_данных" Тогда
         Сканер = Парам.Получить("Объект");
```

```
            Флаг   = Парам.Получить("Флаг");
            Рез = ПосылкаДанных(Сканер, Флаг);
         ИначеЕсли Процесс = "очистить_вход" Тогда
            Сканер = Парам.Получить("Объект");
            Рез    = ОчиститьВход(Сканер);
         Иначе
            Рез = 0;
         КонецЕсли;
      КонецЕсли;
      Форма.Параметр.Установить("ОписаниеРезультата",
                                                ОписаниеРезультата);
      Форма.Параметр.Установить("Результат", Рез);
КонецПроцедуры // ПриОткрытии
```

Обмен данными

После того как сканер подключен и готов к работе, система переходит в состояние ожидания события — посылка данных от внешнего устройства. Обрабатывает событие предопределенная процедура `ОбработкаВнешнегоСобытия()` (рис. 6.4).

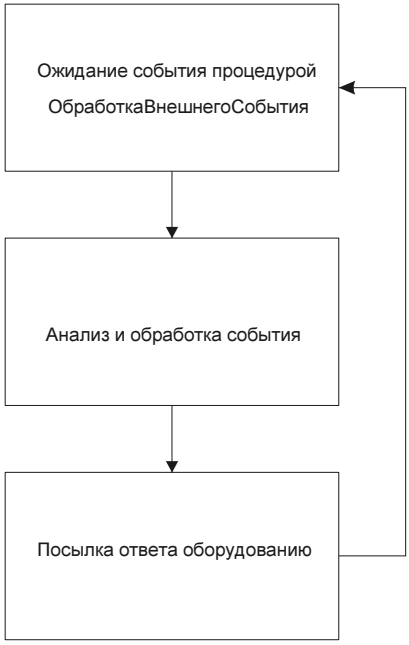


Рис. 6.4. Алгоритм обработки внешнего события

Эта процедура определяется в модуле той формы, в которой используются переданные от устройства данные. При этом следует обратить внимание на то, что обработкой переданных данных будет заниматься та форма, которая будет открыта в момент их передачи. Если в момент передачи данных не было открыто ни одной формы, то событие обработает предопределенная процедура `ОбработкаВнешнегоСобытия()` глобального модуля. Этой процедуре передается три параметра:

- `Источник` — идентификатор модели оборудования;
- `Событие` — идентификатор события;
- `Данные` — собственно данные, переданные оборудованием.

Добавим в наш справочник "Квартиры" реквизит **ШтрихКод** типа "Строка" длиной 13 символов и с признаком **Сортировка** (рис. 6.5).

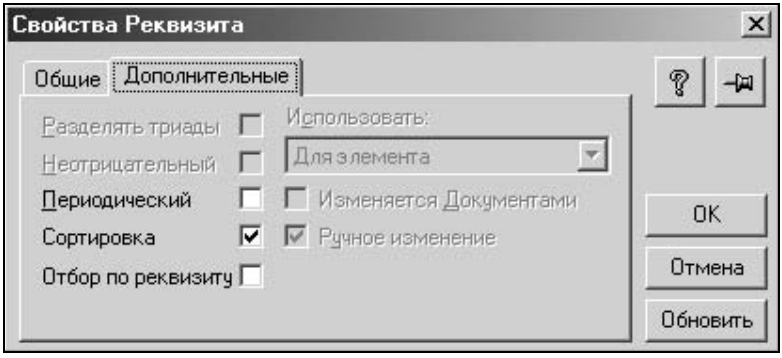


Рис. 6.5. Диалоговое окно свойств реквизита **ШтрихКод** на вкладке **Дополнительные**

В извещении на оплату добавим изображение штрих-кода в обеих секциях **Извещение** и **Квитанция**. При вводе оплаченных корешков извещений на оплату будем использовать сканер штрих-кода для быстрой идентификации квартир. Добавим в нашу конфигурацию документ — "Оплата", в котором реквизит **Квартира** будет иметь тип "Справочник.Квартира" (рис. 6.6).

В документе "Оплата" определим предопределенную процедуру `ОбработкаВнешнегоСобытия()`, которая разбирает штрих-код, считанный сканером, и заполняет строки документа:

```
Процедура ОбработкаВнешнегоСобытия(Источник,Событие,Данные)
  Перем ВремКвартира;
  Если Событие = "BarCodeValue" Тогда
     Если Форма.ТолькоПросмотр() = 0 Тогда
        Штрих-код = СокрЛП(Данные);
```

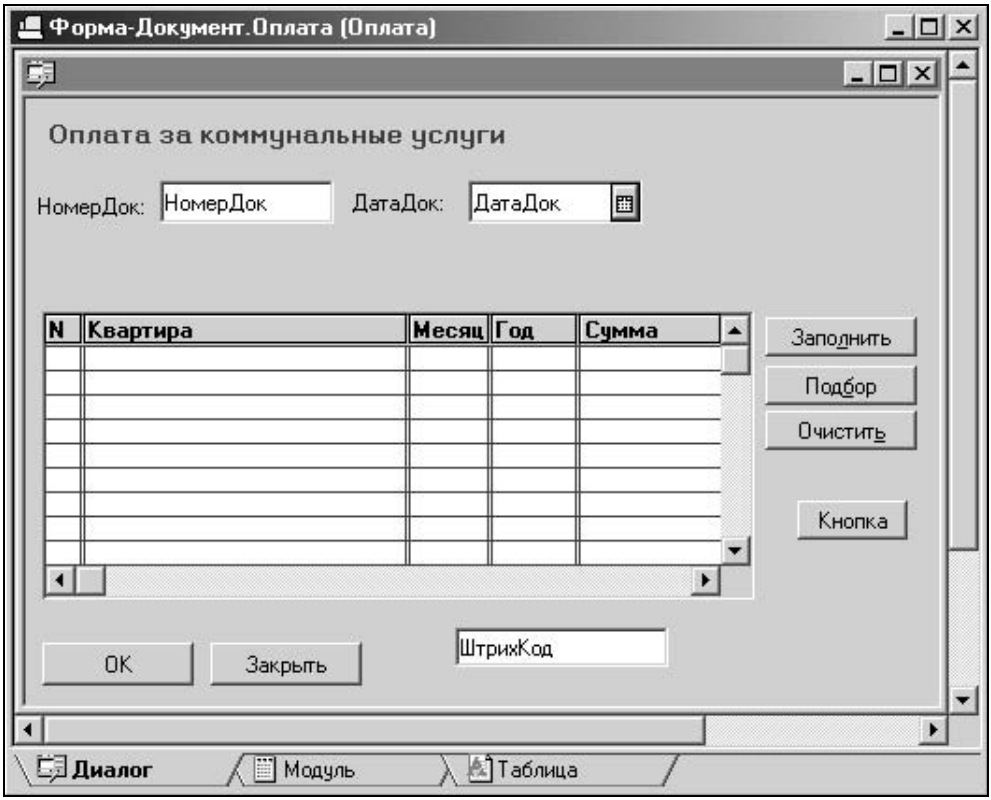


Рис. 6.6. Диалоговое окно документа **Оплата**

```
// Определение типа штрих-кода
        ПрефиксШтрих-кода = глСканерПрефикс;
        Если ПустоеЗначение(ПрефиксШтрих-кода) = 0 Тогда
           Если Найти(ПрефиксШтрих-кода, Лев(Штрих-код, 1)) <> 0 Тогда
// Удаляем спец. символ из штрих-кода
              Штрих-код = Сред(Штрих-код, 2);
           Иначе
              Сообщить
              | ("Формат считанного штрих-кода
              | не предназначен для квартир");
              Возврат 0;
           КонецЕсли;
        КонецЕсли;
// Идентификация полученного штрих-кода
// Используется основной штрих-код квартиры
        ОбКвартира = СоздатьОбъект("Справочник.Квартиры");
```

```
        Рез = ОбКвартира.НайтиПоРеквизиту("ШтрихКод", Штрих-код, 1);
        Если Рез = 1 Тогда
           НоваяСтрока();
           Квартира =  ОбКвартира.ТекущийЭлемент();
        Иначе
           Сообщить("Квартира со штрих-кодом " + Штрих-код + " не
                                                          найдена.");
        КонецЕсли;
     КонецЕсли;
// Обработка закончена. Готовы к получению нового штрих-кода.
     глСканерПосылкаДанных(1);
  КонецЕсли;

            КонецПроцедуры // ОбработкаВнешнегоСобытия()
```

Из приведенного текста процедуры `ОбработкаВнешнегоСобытия()` видно, что процедуре передается три параметра:

- `Источник` — наименование оборудования, например, `"A2000OnLine"` (контрольно-кассовая машина);
- `Событие` — наименование события, например, `BarCodeValue` (получен штрих-код);
- `Данные` — полученные данные.

Процедура `глСканерПосылкаДанных()` определяется в глобальном модуле и передает сканеру ответ о том, что данные обработаны:

```
Функция глСканерПосылкаДанных(Флаг) Экспорт
  Рез = 0;
  Если глСканерВкл = 1 Тогда
     Параметры = СоздатьОбъект("СписокЗначений");
     Параметры.Установить("Процесс", "посылка_данных");
     Параметры.Установить("Флаг"   , Флаг);
     Параметры.Установить("Объект" , глСканер);
     Если ПустоеЗначение(глСканерОбработка) = 1 Тогда
// внутренняя обработка
Если Метаданные.Обработка("ОбслуживаниеСканер" ).Выбран() = 1 Тогда
           ОткрытьФормуМодально("Обработка.ОбслуживаниеСканер",
                                Параметры);
        Иначе
           Сообщить("Отсутствует обработка обслуживания");
        КонецЕсли;
```

```
      Иначе
// внешняя обработка
Если ФС.СуществуетФайл(глКаталогОписаний + глСканерОбработка) = 0 Тогда
            Сообщить("Отсутствует обработка обслуживания");
         Иначе
            ОткрытьФормуМодально("Отчет", Параметры,
                                 глКаталогОписаний + глСканерОбработка);
         КонецЕсли;
      КонецЕсли;
   КонецЕсли;
   Возврат Рез;
КонецФункции // глСканерПосылкаДанных
```

Определим процедуру `ОбработкаВнешнегоСобытия()` в гдобальном модуле:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные)
//  Эта процедура в глобальном модуле отлавливает данные от сканера
//  штрих-кода,
//  когда не открыта ни одна из форм, использующих сканер в своей
//  работе. Пришедшие данные в данном случае пропускаются, чтобы не
//  заполнять буфер сканера
   Если Событие = "BarCodeValue" Тогда
      Если глСканерВключен() = 1 Тогда
         глСканерПосылкаДанных(1);
      КонецЕсли;
   Иначе
      глОбработкаВнешнегоСобытия(Источник, Событие, Данные);
   КонецЕсли;
КонецПроцедуры // ОбработкаВнешнегоСобытия()
```

Глава 7

Рекомендации по сопровождению программы "1С:Предприятие"

Восстановление базы данных

В данной главе речь пойдет о методах восстановления базы данных в тех случаях, когда не было несанкционированных вмешательств со стороны некомпетентного пользователя, т. е. экспериментальных перемещений файлов базы данных. Искажение базы данных происходит в основном по причине аварийного завершения программы и выхода из нее на одной из рабочих станций, находящейся в режиме разделенного доступа. Ничего страшного здесь не происходит, если сразу переиндексировать базу данных. Возможно, ничего не произойдет, если переиндексировать базу позже. Все зависит от того, в какой момент произошел аварийный выход из программы. При нестабильной работе локальной сети такие ситуации возникают часто. Следовательно, вероятность возникновения нарушений целостности базы данных, по данной причине, будет высокой.

Некорректность бухгалтерских итогов

Признаком искажения бухгалтерских итогов является то обстоятельство, что входящее сальдо по счету в отчете "Оборотно-сальдовая ведомость" не совпадает с исходящим сальдо за предыдущий период. Не пытайтесь исправить таблицу бухгалтерских итогов *(см. гл. 1, разд. Файлы компоненты "Бухгалтерский учет")* непосредственно в файле. Скорее всего, вы окончательно испортите таблицу, восстановить которую будет невозможно. Для правильного восстановления таблицы бухгалтерских итогов необходимо выполнить следующие действия:

1. В меню программы "1С:Предприятие" выбрать пункт **Операции**, затем **Управление бухгалтерскими итогами** (рис. 7.1).
2. В диалоговом окне **Управление бухгалтерскими итогами** в поле **по:** необходимо задать период, предшествующий началу работы в программе бухгалтерии, т. е. период, предшествующий вводу начальных остатков (рис. 7.2).

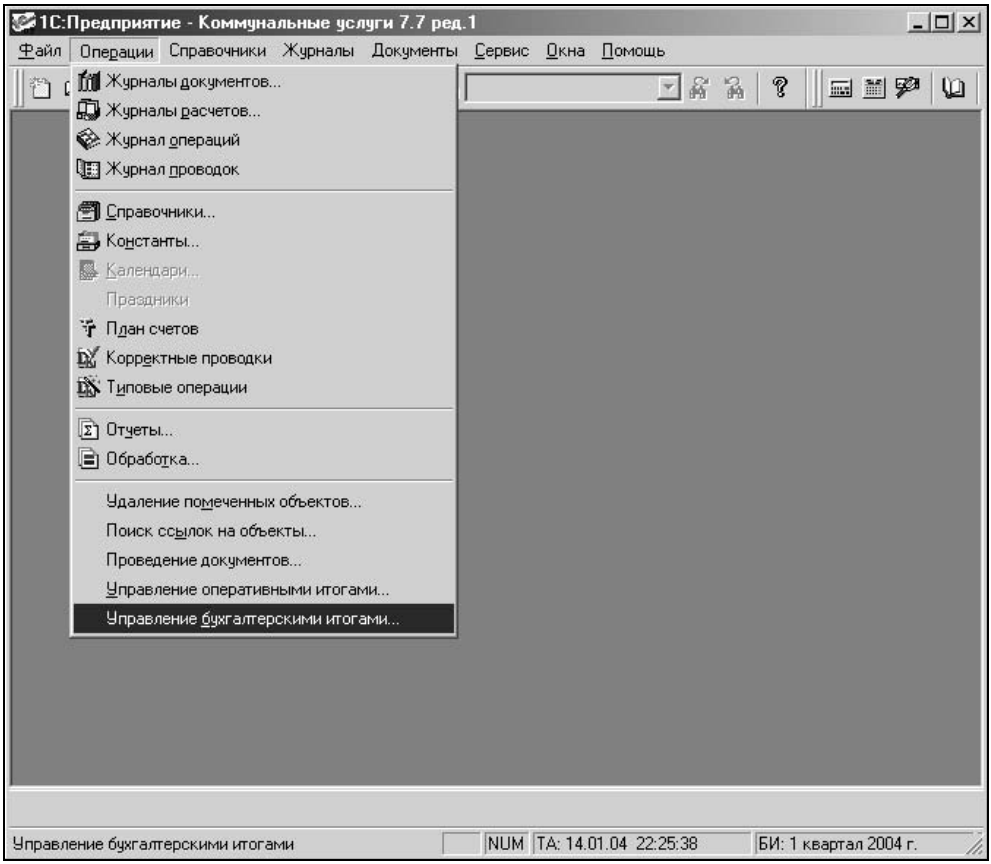


Рис. 7.1. Меню **Операции**

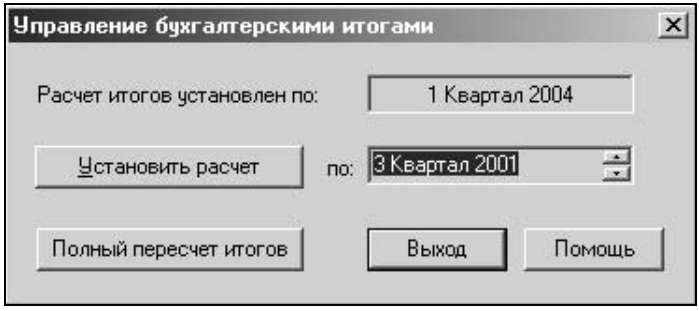


Рис. 7.2. Диалоговое окно **Управление бухгалтерскими итогами**

3. Нажмите кнопку **Установить расчет**. После смены периода в поле **по:** нажмите снова указанную кнопку и т. д., пока не восстановится текущий период в поле **Расчет итогов установлен по:**.

Внимание

Простое нажатие кнопки **Полный пересчет итогов** к восстановлению таблицы бухгалтерских итогов не приведет.

Тестирование и исправление информационной базы

Возникновение документов и операций без номеров или дат создания, исчезновение документов, при условии присутствия их операций, исчезновение ссылок на объекты и т. п. свидетельствует о нарушении целостности иформационной базы (ИБ) данных и требует ее восстановления. Иногда нарушения носят фатальный характер. В таких случаях ИБ не загружается. Для устранения подобных ситуаций в Конфигураторе предусмотрен режим восстановления ИБ.

Для запуска режима восстановления необходимо выполнить следующие действия:

1. В меню Конфигуратора выбрать пункт **Администрирование**, затем пункт **Тестирование и исправление ИБ** (см. рис. 7.3).

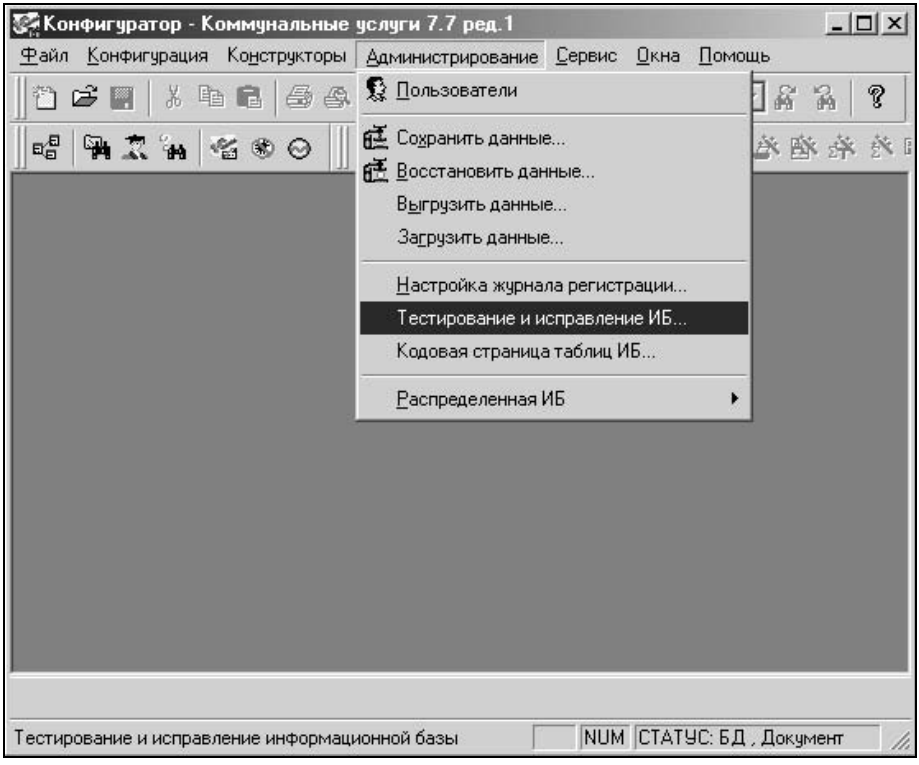


Рис. 7.3. Меню **Администрирование**

Диалог **Тестирование и исправление информационной базы** служит для установки параметров тестирования информационной базы:

- выбор этапов тестирования;
- выбор режима тестирования (см. рис. 7.4).

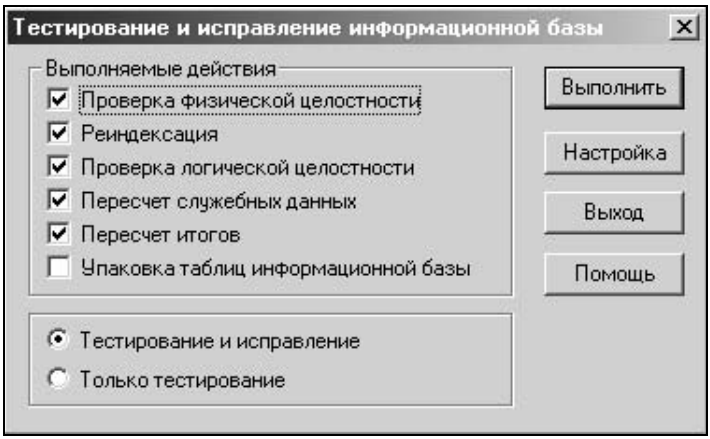


Рис. 7.4. Диалоговое окно
Тестирование и исправление информационной базы

2. Задать параметры тестирования и исправления информационной базы. Рекомендуется перед исправлением информационной базы провести ее тестирование. По результатам тестирования выбираются способы исправления ИБ, которые задаются в диалоговом окне **Настройка исправления информационной базы** (рис. 7.5), вызываемого нажатием кнопки **Настройка** (см. рис. 7.4). По умолчанию установлены все этапы тестирования, кроме упаковки таблиц информационной базы. Установки можно изменить. Этапы тестирования допускается производить независимо друг от друга, устанавливая любые флажки. Правда, в том случае, если восстановление физической целостности базы нарушило ее индексную структуру, реиндексация будет проведена независимо от того, установлен ли соответствующий флажок или нет. Если предполагается провести только тестирование информационной базы, то выбирается опция **Только тестирование**, если же требуется еще и исправление ИБ, то отмечается опция **Тестирование и исправление** (рис. 7.4).
3. Задать способы исправления информационной базы в диалоговом окне **Настройка исправления информационной базы** нажатием кнопки **Настройка** (рис. 7.5).

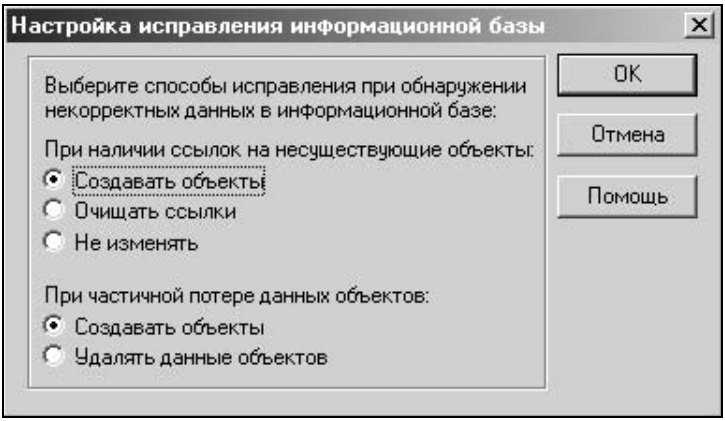


Рис. 7.5. Диалоговое окно
Настройка исправления информационной базы

Ошибка запуска Конфигуратора

Ошибка запуска типа "RunTime Error" возникает по нескольким причинам. К данной ошибке приводят частые аварийные выходы из программы, особенно, если выход происходит на одной рабочей станции. Программа переиндексирует файлы только в монопольном режиме. При нестабильной работе локальной сети иногда возникает ситуация, когда режим блокировки какого-либо файла не снимается — атрибуты доступа не соответствуют допустимым параметрам запуска программы. В таких случаях не запускается даже Конфигуратор. Для устранения данной неисправности необходимо скопировать весь каталог базы данных в другое место и переназначить путь к нему. Как правило, атрибуты доступа при копировании восстанавливаются. Иногда презагрузка системы приводит к устранению ошибки.

Ошибка открытия файлов

Довольно распространенной является ошибка открытия файла. В этом случае, при всех внешних признаках "благополучия" файла, он все же может не открываться. Причиной этого является недостаток ресурсов компьютера, на котором расположена ИБ. Как правило, критическим ресурсом является количество одновременно открытых файлов. Чаще всего это происходит в системе Windows 98, у которой данный ресурс слабый. Замена системы на Windows 2000 Server устраняет ошибку, т. к. данный ресурс в ней в два раза выше.

Обновление конфигурации

Для осуществления обновлений конфигурации, поставляемых фирмой 1С, с сохранением данных в информационной базе, рекомендуется режим объединения конфигураций, который позволяет программисту принимать решения по замене старых объектов на новые. Главным достоинством данного режима является то, что после анализа отчета по обновлению можно внести корректировки или вовсе отказаться от объединения.

Для обновления конфигурации в режиме объединения необходимо выполнить следующие действия:

1. В меню Конфигуратора выбрать пункт **Конфигурация | Объединение конфигураций** (рис. 7.6).

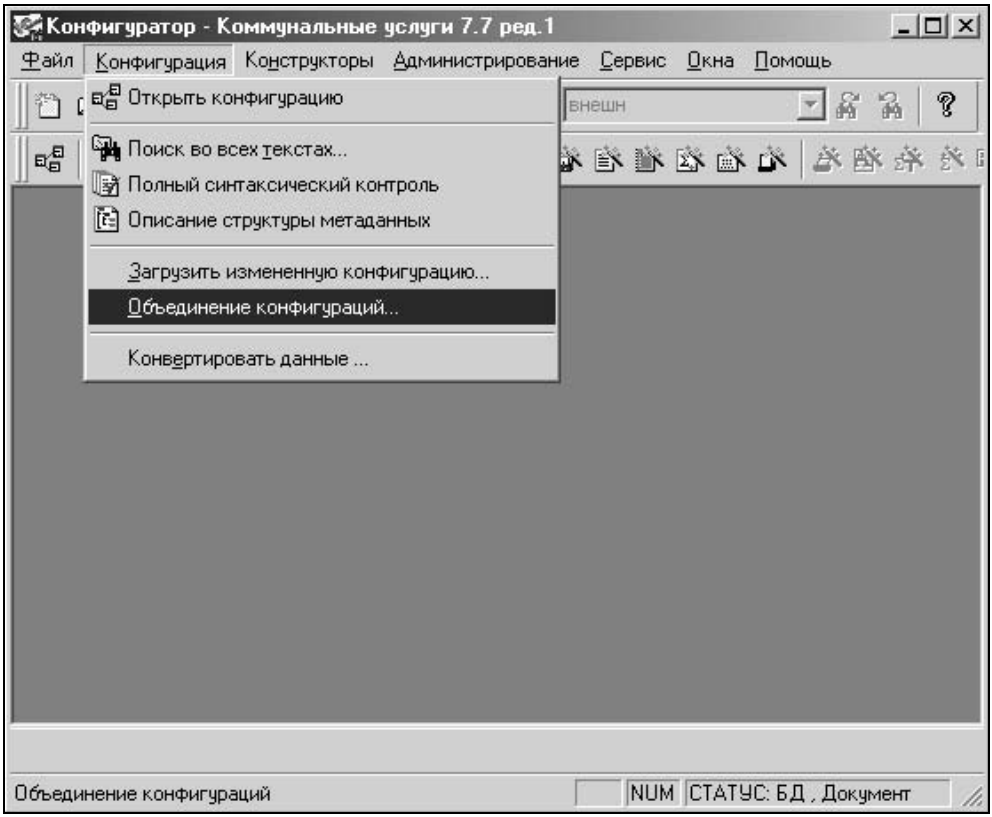


Рис. 7.6. Меню **Конфигурация**

2. Задать в диалоговом окне **Открыть файл конфигурации** (рис. 7.7) файл конфигурации, с которой следует объединить текущую конфигурацию (1cv7.md).

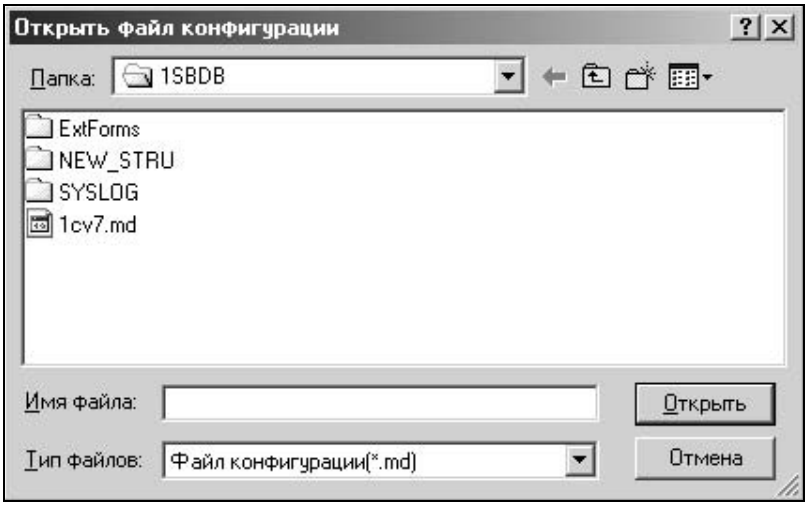


Рис. 7.7. Диалоговое окно выбора файла конфигурации для объединения

3. В окне **Объединение конфигураций** установить те признаки объектам, которые будут участвовать в обновлении (рис. 7.8).

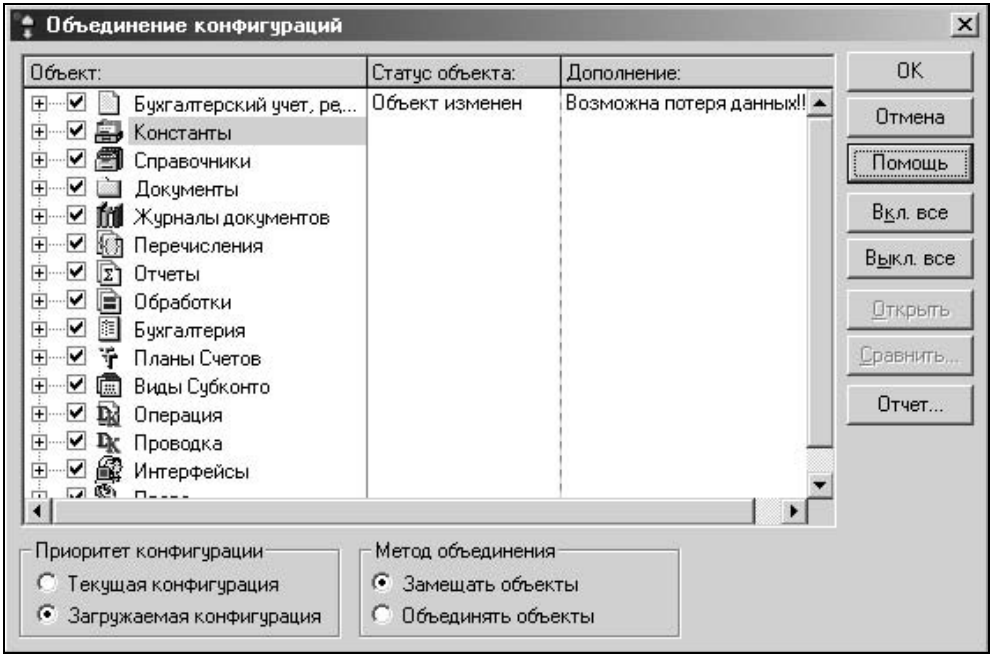


Рис. 7.8. Диалоговое окно **Объединение конфигураций**

В окне обновления (см. рис. 7.8) показаны объекты метаданных, по которым найдены какие-либо изменения. Установленный флажок определяет, что данные объекты двух конфигураций будут объединяться. Те объекты, в строке которых текст написан серым, не могут быть самостоятельно включены в объединение конфигураций, так как являются элементами других (агрегатных) объектов. Элементы агрегатного объекта могут быть включены или выключены вместе с самим объектом. Колонка **Статус объекта**, для конкретного объекта, может содержать указание — **Объект изменен**, **Объект добавлен** или не содержать никакого указания. Последнее означает, что различий непосредственно по данному объекту метаданных, в сравниваемых конфигурациях, не найдено. То есть данный объект — агрегатный и изменения найдены для какого-то из составляющих его элементов. Чтобы обнаружить измененный или добавленный элемент, нужно развернуть соответствующую ветвь дерева. Колонка **Дополнение**, для некоторых объектов, содержит сообщение — **Возможна потеря данных!!!**. Оно выдается, если есть вероятность, что изменения могут привести к потере данных (при этом имеется в виду потеря данных в текущей конфигурации, а не в загружаемой).

4. Задать **Метод объединения**. При выборе метода **Замещать объекты** объект метаданных будет добавлен, если он новый, или замещен, если он измененный. В случае добавления переносится вся структура объекта, модули, описания, формы. При замещении сохраняется объект той конфигурации, у которой установлен приоритет. При выборе метода **Объединять объекты**, элементы объектов, имеющиеся только в одной из объединяемых конфигураций, сохраняются в объединенной конфигурации. Отличающиеся элементы выбираются в зависимости от установленного приоритета.
5. Задать **Приоритет конфигурации**. **Текущая конфигурация** выбирается в том случае, если из загружаемой конфигурации необходимо взять только новое и максимально сохранить старое. Отличающиеся элементы добавляются в виде комментария (текст) или с измененным названием (таблица). При выборе приоритета **Загружаемая конфигурация** отличающиеся элементы текущей конфигурации превращаются в комментарий (текст) или им изменяется название (таблица).
6. Просмотреть изменения в отчете, сформированном с помощью кнопки **Отчет** (в случае необходимости — скорректировать установки).
7. Запустить процесс объединения нажатием кнопки **ОК**.
8. После объединия конфигураций новая конфигурация еще не записана. В этом случае, перед сохранением изменений, будет предложено последнее предупреждение для принятия решения (рис. 7.9).

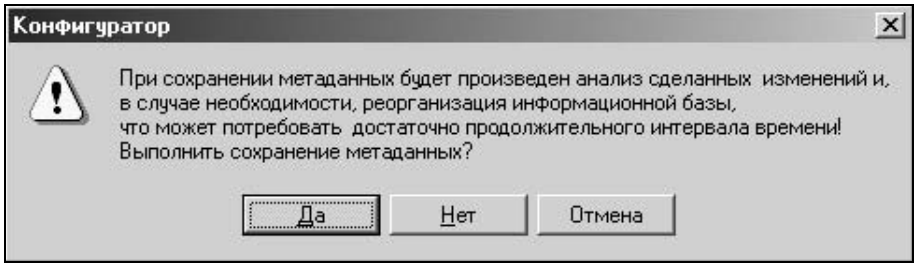


Рис. 7.9. Диалоговое окно подтверждения изменений ИБ

Методические рекомендации по изменению конфигурации

Обновление конфигурации, периодически поставляемое фирмой 1С, приводит к проблеме потери ранее разработанных изменений для тех объектов метаданных, которые предусмотрены в штатном составе конфигурации. На первый взгляд, самое простое решение проблемы — скопировать объект метаданных, переименовать его, модифицировать и включить в соответствующий журнал и пункт меню. Однако такие объекты стандартным механизмом обновления не затрагиваются. Но не все так просто на самом деле. Во многих модулях ведется анализ вида документа, не говоря уже о предназначении справочника. Поскольку новый вид документа в штатной конфигурации не предусмотрен, то и из анализа он, грубо говоря, выпадает. Можно вставить в анализ вида документа вновь созданный документ, но тогда придется следить за обновлением уже этого объекта. Можно отследить всю цепочку и связанных таким образом объектов, но что делать с глобальным модулем?

Внимание

Категорически не рекомендуется отказываться от обновления глобального модуля. Изменения в служебных функциях глобального модуля не описываются в инструкциях по обновлению, а используются они повсеместно.

Таким образом, данная методика хороша при условии, что вновь созданный документ не должен участвовать в анализе вида документа в глобальном модуле. Например, при заполнении счета-фактуры на основании другого документа ведется анализ:

```
Если ДокОсн.Вид()="ПередачаОС" Тогда

// Табличная часть счета-фактуры будет заполнена из справочника
// "Основные средства", если ввод на основании документа "ПередачаОС"

        …

ИначеЕсли ДокОсн.Вид()="ПередачаНМА" Тогда
```

```
// Табличная часть счета-фактуры будет заполнена из справочника
// "Нематериальные активы", если ввод на основании документа
// "ПередачаНМА"
      …
ИначеЕсли ДокОсн.Вид()="ОтпускМатериаловНаСторону" Тогда
// Табличная часть счета-фактуры будет заполнена из справочника
// "Материалы", если ввод на основании документа
// "ОтпускМатериаловНаСторону"
      …
ИначеЕсли (ДокОсн.Вид()="ОказаниеУслуг") или
          (ДокОсн.Вид()="РасходнаяНакладная") или
          (ДокОсн.Вид()="РеализацияОтгруженнойПродукции") или
          (ДокОсн.Вид()="ВыполнениеЭтапаРабот") Тогда
// Табличная часть счета-фактуры будет заполнена из справочника
// "Номенклатура", если ввод на основании документов
// "ОказаниеУслуг", "РасходнаяНакладная",
// "РеализацияОтгруженнойПродукции", "ВыполнениеЭтапаРабот")
      …
ИначеЕсли (ДокОсн.Вид() = "Выписка") или
                    (ДокОсн.Вид() = "ПриходныйОрдер") Тогда
// Табличная часть счета-фактуры будет заполнена на аванс,
//если ввод на основании документов " Выписка", "ПриходныйОрдер"
      …
КонецЕсли;
```

Базовая функция в глобальном модуле будет иметь следующий вид (здесь `Конт` — контекст вызывающей формы):

```
Процедура глДляЗаполненияКнигиПокупок(Конт) Экспорт
  Если Конт.Договор.ОплатаДоговора = 2 Тогда
     Возврат;
  КонецЕсли;
  БезНДС20=0;
  БезНДС10=0;
  НДС20=0;
  НДС10=0;
  НДС0=0;
  Освобождаемые=0;
  Если (Конт.Вид() = "УслугиСтороннихОрганизаций") или
       (Конт.Вид() = "ПоступлениеТоваров") или
       (Конт.Вид() = "ПоступлениеМатериалов") или
       (Конт.Вид() = "ПоступлениеОборудования") или
       (Конт.Вид() = "ПоступлениеОС") или
```

```
      (Конт.Вид() = "ПоступлениеНМА") Тогда
    Если Конт.НДСвключатьВСтоимость = 1 Тогда
       Возврат;
    КонецЕсли;
    Курс = ?(Конт.Курс=0,1,Конт.Курс);
    Всего = (Конт.Итог("Всего") - Конт.Итог("НП"))*Курс;
    Конт.ВыбратьСтроки();
    Пока Конт.ПолучитьСтроку() = 1 Цикл
       Если Конт.Вид() = "ПоступлениеТоваров" Тогда
         Если Конт.Товар.ТипТовара = Перечисле-
         ние.ТипыТоваров.НаКомиссии Тогда
            Продолжить;
         КонецЕсли;
       КонецЕсли;
       СуммаБезНалогов = (Конт.Всего - Конт.НДС - Конт.НП)*Курс;
       Если СуммаБезНалогов > 0 Тогда
        СтавкаНДС = 100*Конт.НДС/(Конт.Всего - Конт.НДС - Конт.НП);
       Иначе
          СтавкаНДС = 20;
       КонецЕсли;
       Если СтавкаНДС>10.5 Тогда
          НДС20=НДС20+Конт.НДС*Курс;
          БезНДС20=БезНДС20+СуммаБезНалогов;
       ИначеЕсли СтавкаНДС>0 Тогда
          НДС10=НДС10+Конт.НДС*Курс;
          БезНДС10=БезНДС10+СуммаБезНалогов;
       Иначе
          Освобождаемые=Освобождаемые+СуммаБезНалогов;
       КонецЕсли;
    КонецЦикла;
 ИначеЕсли Конт.Вид() = "СчетФактураПолученный" Тогда
    Курс = ?(Конт.Курс=0,1,Конт.Курс);
    Всего = Конт.Всего*Курс;
    БезНДС20=Конт.СуммаБезНДС20*Курс;
    БезНДС10=Конт.СуммаБезНДС10*Курс;
    НДС20=Конт.НДС20*Курс;
    НДС10=Конт.НДС10*Курс;
    Освобождаемые=Конт.СуммаСовсемБезНДС*Курс;
```

```
    Если Конт.НДСпоСтавкеНольПроцентов = 1 Тогда
      НДС10 = 0;
      НДС20 = 0;
      БезНДС10 = 0;
      БезНДС20 = 0;
      НДС0 = Всего;
      КонецЕсли;
КонецЕсли;
Если БезНДС20>0 Тогда
  Конт.Операция.НоваяПроводка();
  Конт.Операция.НомерЖурнала = "АВ";
  Конт.Операция.ПервичныйДокумент =
                        глПредставлениеПервичногоДокумента (Конт);
  Конт.Операция.Кредит.Счет = СчетПоКоду("ЗПК.20.Б");
  Конт.Операция.Кредит.Субконто(1,Конт.Контрагент);
  Конт.Операция.Кредит.Субконто(2,Конт.Договор);
  Конт.Операция.Кредит.Субконто(3,Конт.ТекущийДокумент());
  Конт.Операция.Сумма = БезНДС20;
  Конт.Операция.СодержаниеПроводки = "Данные для автоматического
                                    учета НДС";
КонецЕсли;
Если НДС20>0 Тогда
  Конт.Операция.НоваяПроводка();
  Конт.Операция.НомерЖурнала = "АВ";
  Конт.Операция.ПервичныйДокумент  =
                      глПредставлениеПервичногоДокумента (Конт);
  Конт.Операция.Кредит.Счет = СчетПоКоду("ЗПК.20.Н");
  Конт.Операция.Кредит.Субконто(1,Конт.Контрагент);
  Конт.Операция.Кредит.Субконто(2,Конт.Договор);
  Конт.Операция.Кредит.Субконто(3,Конт.ТекущийДокумент());
  Конт.Операция.Сумма = НДС20;
  Конт.Операция.СодержаниеПроводки =
                     "Данные для автоматического учета НДС";
КонецЕсли;
Если БезНДС10>0 Тогда
  Конт.Операция.НоваяПроводка();
  Конт.Операция.НомерЖурнала = "АВ";
  Конт.Операция.ПервичныйДокумент =
                        глПредставлениеПервичногоДокумента (Конт);
```

```
    Конт.Операция.Кредит.Счет = СчетПоКоду("ЗПК.10.Б");
    Конт.Операция.Кредит.Субконто(1,Конт.Контрагент);
    Конт.Операция.Кредит.Субконто(2,Конт.Договор);
    Конт.Операция.Кредит.Субконто(3,Конт.ТекущийДокумент());
    Конт.Операция.Сумма = БезНДС10;
    Конт.Операция.СодержаниеПроводки =
                          "Данные для автоматического учета НДС";
КонецЕсли;
Если НДС10>0 Тогда
    Конт.Операция.НоваяПроводка();
    Конт.Операция.НомерЖурнала = "АВ";
    Конт.Операция.ПервичныйДокумент  =
                       глПредставлениеПервичногоДокумента (Конт);
    Конт.Операция.Кредит.Счет = СчетПоКоду("ЗПК.10.Н");
    Конт.Операция.Кредит.Субконто(1,Конт.Контрагент);
    Конт.Операция.Кредит.Субконто(2,Конт.Договор);
    Конт.Операция.Кредит.Субконто(3,Конт.ТекущийДокумент());
    Конт.Операция.Сумма = НДС10;
    Конт.Операция.СодержаниеПроводки =
                          "Данные для автоматического учета НДС";
КонецЕсли;
Если НДС0>0 Тогда
    Конт.Операция.НоваяПроводка();
    Конт.Операция.НомерЖурнала = "АВ";
    Конт.Операция.ПервичныйДокумент =
                      глПредставлениеПервичногоДокумента (Конт);
    Конт.Операция.Кредит.Счет = СчетПоКоду("ЗПК.0");
    Конт.Операция.Кредит.Субконто(1,Конт.Контрагент);
    Конт.Операция.Кредит.Субконто(2,Конт.Договор);
    Конт.Операция.Кредит.Субконто(3,Конт.ТекущийДокумент());
    Конт.Операция.Сумма = НДС0;
    Конт.Операция.СодержаниеПроводки =
                          "Данные для автоматического учета НДС";
КонецЕсли;
Если Освобождаемые>0 Тогда
    Конт.Операция.НоваяПроводка();
    Конт.Операция.НомерЖурнала = "АВ";
    Конт.Операция.ПервичныйДокумент  =
                       глПредставлениеПервичногоДокумента (Конт);
```

```
    Конт.Операция.Кредит.Счет = СчетПоКоду("ЗПК.БН");
    Конт.Операция.Кредит.Субконто(1,Конт.Контрагент);
    Конт.Операция.Кредит.Субконто(2,Конт.Договор);
    Конт.Операция.Кредит.Субконто(3,Конт.ТекущийДокумент());
    Конт.Операция.Сумма = Освобождаемые;
    Конт.Операция.СодержаниеПроводки =
                          "Данные для автоматического учета НДС";
КонецЕсли;
КонецПроцедуры
```

Аналогичная ситуация с книгой продаж.

Из вышеизложенного следует, что лучше избегать включения в глобальный модуль своих процедур, функций и переменных. Если этого избежать невозможно, придется модифицировать глобальный модуль после каждого обновления релиза.

Предметный указатель

А

Активные пользователи конфигурации в сети 16
Анализ и исправление ошибок 287

Б

Бухгалтерские итоги 35

Г

Глобальный модуль 109

Д

Директива 69

Ж

Журналы 19
Журналы расчетов 22

И

Информация о текущей дате и времени 16

К

Компонента 22, 24
Константы 17
Конфигуратор 1

М

Метод:
- WindowsКаталог() 270
- АвтоСохранение() 275
- Актуальность() 178
- АтрибутыФайла() 269
- Блокировка() 152
- ВертикальноеПоложение() 256
- Вид() 148
- ВидимостьКолонки() 241
- ВключатьПодчиненные() 176
- ВключатьСубсчета() 212
- ВКонце() 273
- Восстановить() 275
- ВременныйРасчет() 185
- ВставитьСтроку() 282
- Выбран() 151
- Выбрана() 191
- ВыбранаПоДт() 225
- ВыбранаПоКт() 225
- Выбрать() 159
- ВыбратьВалюты() 218
- ВыбратьДвижения() 182
- ВыбратьДвижения Документа() 179
- ВыбратьДокументы() 162
- ВыбратьИтоги() 184
- ВыбратьКаталог() 267
- ВыбратьКорСубконто() 222
- ВыбратьКорСчета() 217
- ВыбратьОперации() 186

(продолжение рубрики см. на стр. 330)

Метод *(продолжение)*:
ВыбратьОперации СПроводками() 192, 193
ВыбратьПериоды() 219
ВыбратьПодчиненные Документы() 163
ВыбратьПоЗначению() 164, 194
ВыбратьПоНомеру() 166
Выбрать ПоПоследовательности() 166
ВыбратьСтроки() 235, 285
ВыбратьСтроку() 236
ВыбратьСубконто() 220
ВыбратьСчета() 216
ВыбратьФайл() 266
ВыбратьФайл Картинки() 267
ВыбратьЭлементы() 172
ВыбратьЭлементы ПоРеквизиту() 173
Вывести() 250
ВывестиСекцию() 251
Выгрузить() 240
ВыполнитьЗапрос() 214
ВысотаСекции() 252
Горизонтальное Положение() 256
ДО() 203 226
Добавить() 275
ДобавитьИндекс() 277
ДобавитьПоле() 277
ДобавитьСтроку() 282
Загрузить() 240
ЗакрытьФайл() 272, 284
ЗаменитьСтроку() 283
Записать() 158, 245, 275, 284
ЗаписатьСтроку() 284
ЗаписьУдалена() 275
Заполнить() 239
Защита() 244
ЗначениеТекущейЯчейки() 249
ИспользоватьВладельца() 175
Использовать КорСубконто() 194, 213
ИспользоватьПланСчетов() 196
Использовать РазделительУчета() 196
ИспользоватьРодителя() 175
Использовать Субконто() 193, 213
ИспользоватьФормат() 249
ИсходнаяТаблица() 249
Итог() 239
КО() 204, 226
КодоваяСтраница() 277, 283, 285
КодОшибки() 278
КоличествоЗаписей() 275
КоличествоИндексов() 276
КоличествоКолонок() 231
КоличествоПолей() 275
КоличествоСтрок() 239, 281
КоличествоЭкземпляров() 248
КонецПериодаПоДате() 261
КонецТекущегоПериода() 261
КонПериода() 199
Контроль() 257
КопироватьФайл() 268
КорСубконто() 223
Курсив() 256
НазначитьТип() 143
Найти() 273
НайтиДокумент() 162
НайтиДокументПоНомеру() 162
НайтиЗначение() 237
НайтиОперацию() 189
НайтиПервыйФайл() 269
НайтиПоКлючу() 274
НайтиПоКоду() 169
НайтиПоНаименованию() 170
НайтиПореквизиту() 171
НайтиСледующийФайл() 269
НайтиЭлемент() 168
Напечатать() 248
НачалоПериодаПоДате() 261
НачалоТекущегоПериода() 261
НачПериода() 199
Новая() 156
НоваяГруппа() 177

НоваяКолонка() 231
НоваяПроводка() 157
НоваяСтрока() 235
Новый() 156
НомерЗаписи() 273
НомерПоля() 276
НомерСтроки() 184
ОБ() 204
Область() 252
ОбластьПечати() 246
ОбратныйПорядок() 161
Объединить() 255
ОписаниеИндекса() 276
ОписаниеПоля() 276
Опции() 212, 245
ОсновныеИтоги() 196
Открыта() 271
Открыть() 250, 282
ОткрытьФайл() 271, 284
Отменить() 275
Очистить() 239, 244, 275, 283
ОчиститьФайл() 272
ПараметрыСтраницы() 247
Первая() 272
ПереименоватьФайл() 268
Переиндексировать() 272
Перейти() 273
ПериодД() 197
ПериодКВ() 198
ПериодКВН() 198
ПериодМ() 198
ПериодМНГ() 199
ПериодМНК() 198
ПериодПоДате() 261
ПланСчетов() 192
Подчеркнутый() 256
Показать() 244 283
ПоказыватьУдаленные() 272
Полужирный() 255
Получить() 262
ПолучитьАтрибут() 127, 145
ПолучитьВалюту() 219
ПолучитьВремя() 168
ПолучитьДвижение() 184
ПолучитьДокумент() 167
ПолучитьЗначение() 237
ПолучитьЗначениеПоля() 274
ПолучитьИтог() 184
ПолучитьКорСубконто() 223
ПолучитьКорСчет() 218
ПолучитьОперацию() 189
ПолучитьПараметрыКолонки() 234
ПолучитьСекцию() 251
ПолучитьСтроку() 235, 281
ПолучитьСтрокуПоНомеру() 236
ПолучитьСубконто() 221
ПолучитьСчет() 217
ПолучитьЭлемент() 174
ПометкаУдаления() 161
ПорядокКодов() 176
ПорядокНаименований() 177
ПорядокРеквизита() 177
Последняя() 272
ПредставлениеВида() 150
ПредставлениеКорСубконто() 224
ПредставлениеСубконто() 223
Предыдущая() 273
ПрисоединитьСекцию() 251
ПрочитатьСтроку() 285
РазмерБуфера() 285
РазмерШрифта() 255
Рамка() 258
РамкаСверху() 258
РамкаСлева() 258
РамкаСнизу() 258
РамкаСправа() 258
РассчитатьРегистрыНа() 178
РассчитатьРегистрыПо() 178
Расшифровка() 253
Свернуть() 240
СвободноеМестоНаДиске() 270
СдвинутьСтроку() 236
СделатьНеПроведенным() 168
Сжать() 272
СКД() 200, 226
СКДР() 206

(окончание рубрики см. на стр. 332)

Метод *(окончание)*:
СКДРС() 210, 227
СКК() 203, 226
СККР() 206
СККРС() 210
СККРС() 227
Скопировать() 275
Следующая() 272
СНД() 199 225
СНДР() 205
СНДРС() 207, 227
СНК() 200, 225
СНКР() 205
СНКРС() 209, 227
СнятьПометкуУдаления() 161
СоздатьИндексныйФайл() 277
СоздатьКаталог() 270
СоздатьОбъект() 154
СоздатьФайл() 271
Сортировать() 238
Субконто() 223
СуществуетФайл() 268
ТекКаталог() 270
ТекущаяКолонка() 241
ТекущаяСтрока() 240
ТекущийДокумент() 184
ТекущийИндекс() 273
ТекущийПериод() 262
ТолькоПросмотр() 244, 283
Удалить() 160, 275
УдалитьКаталог() 270
УдалитьКолонку() 233
УдалитьСтроки() 235
УдалитьСтроку() 235, 283
УдалитьФайл() 268
Установить() 262
УстановитьАтрибут() 146
УстановитьВремя() 168
УстановитьЗначение() 237
УстановитьЗначениеПоля() 274
УстановитьПараметры Колонки() 233
УстановитьТекущий Период() 261
УстановитьФильтр() 166
УстТекКаталог() 270
Фиксировать() 242
ФиксШаблон() 282
Шаблон() 282
ШиринаСекции() 252
Шрифт() 255
ЭкземпляровНаСтранице() 248
ЭтоГруппа() 225
ВставитьКолонку() 232
Методы:
получения основных итогов бухгалтерских расчетов 197
получения бухгалтерских итогов 196
регистров 177

Н

Нарушение синтаксических конструкций 287

О

Обнаружение синтаксических ошибок 287
Оператор:
Возврат 68
ВызватьИсключение 66
ПопыткаИсключение 63
Прервать 68
Продолжить 67
условного выполнения 59
цикла 62
безусловной передачи управления 66
Описание структуры документа 8
Отчеты по субконто 37

П

Панель:
дерево тем 295
справка 295

Перечень корректных проводок 39
Периодические реквизиты документов 17
План счетов 28
Правила создания процедур 53
Предопределенные процедуры 71
Проводки 32
Процедура:
АрхивироватьДокумент() 91
ВводНаОсновании() 84, 149
ВводНового() 75, 101
ОбработкаВыбора Значения() 101
ОбработкаПодбора() 100
ОбработкаПроведения() 87
ОбработкаУдаления Проведения() 91
ОбработкаЯчейки Таблицы() 118
ПриВводеСтроки() 80, 86
ПриВыбореВладельца() 82, 99
ПриВыбореЗакладки() 100
ПриВыбореРодителя() 82
ПриВыбореСтроки() 80, 92
ПриВыбореСчета() 144
ПриВыбореЯчейки Таблицы() 101
ПриВыклВклПроводок Операции() 117
ПриЗавершенииРаботы Системы() 113
ПриЗакрытии() 79
ПриЗаписи() 80
ПриЗаписи() 75
ПриЗаписиИстории() 115
ПриЗаписиКонстанты() 115
ПриИзмененииВремени Документа() 116
ПриИзмененииПорядка Строк() 86
ПриИсправлении Результата() 94
ПриНачалеВыбора Значения() 101
ПриНачалеРаботыСистемы() 111
ПриНачалеРедактирования Строки() 80, 86
ПриОкончанииРедактирования Строки() 86
ПриОткрытии() 76, 79, 102
ПриОткрытииИстории() 114
ПриОтмененИсправления() 95
ПриОтмененПроведения Документа() 117
ПриПереносеЭлемента ВДругуюГруппу() 81
ПриПовторномОткрытии() 100
ПриРасчете() 95
ПриРедактировании НовойСтроки() 80
ПриСменеИерархии() 82
ПриСменеРасчетного Периода() 118
ПриУдаленииДокумента() 116
ПриУдаленииИстории() 115
ПриУдаленииСтроки() 86
ПриУдаленииСчета (УдалСчетРежим) 117
ПриУдаленииЭлемента() 114
ПриУстановкеГраницы Просмотра() 98
ПриУстановкеИнтервала() 92
ПриУстановкеОтбора() 82, 92, 97
ПриУстановкеОтбора() 117
ПриУстановке Представления() 98

Р

Распределенная база данных 21

С

Справочник 2
Синтаксический контроль запросов 291

Синтакс-Помощник 122, 294
Системные таблицы 15
Словарь
  базы данных 1, 10
Структура
  подчиненности 20

Т

Таблицы:
  операций 31
  метаданных 15
Точка актуальности 16

У

Учет объектов с большим
  количеством операций 44

Ф

Файл конфигурации 1
Функция 56
  ГлКонтрольДаты
    Документа() 150
  Оплата() 164
  Определение Владельца() 154
  СтатусВозврата(Статус) 74