Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Воронежский государственный технический университет»

ОСНОВЫ КОНФИГУРИРОВАНИЯ В СРЕДЕ 1С: ПРЕДПРИЯТИЕ 8.3

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

к лабораторным работам по курсу
«Конфигурирование в системе 1С:»
для студентов направления
09.03.02 Информационные системы и технологии
профиля Информационные системы и технологии цифровизации

Составитель Минаева Ю.В.

Воронеж 2021

Лабораторная работа №1-2
Знакомство с конфигуратором. Константы. Перечисления.

Создание пустой информационной базы

Запустите «1С: Предприятие». В открывшемся диалоге вы увидите список информационных баз (рис. 1).

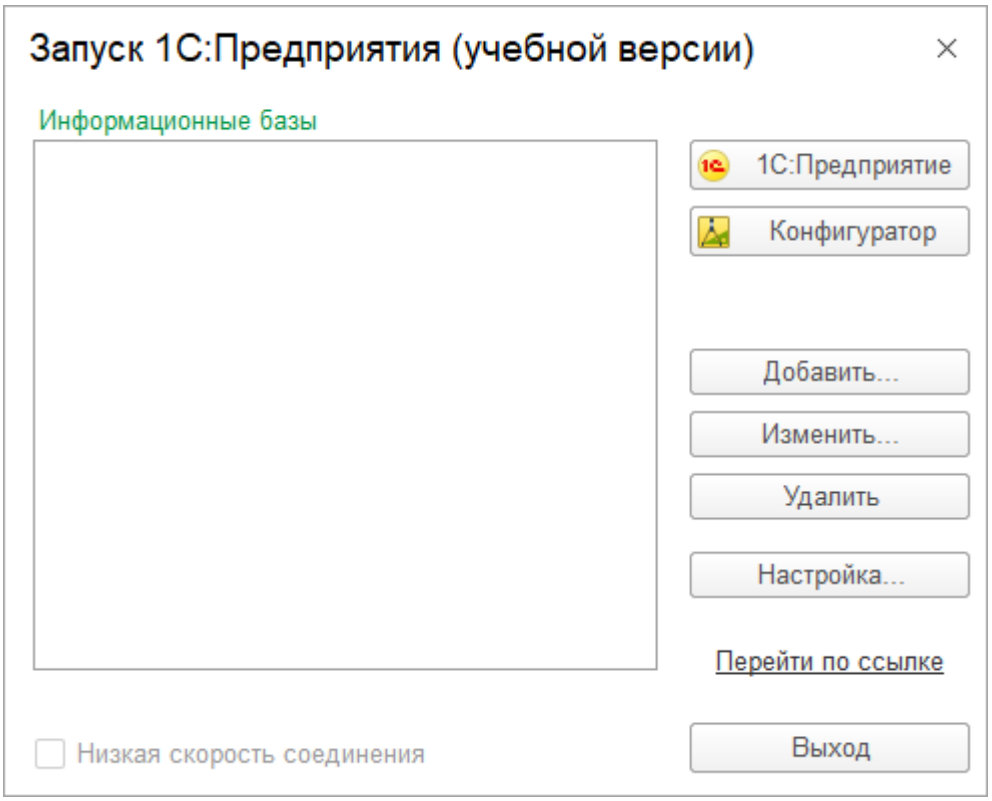


Рисунок 1 – Диалоговое окно создания и редактирования информационных баз

Для создания новой базы надо нажать кнопку «Добавить». В открывшемся диалоге выберите пункт Создание новой информационной базы (рис. 2).

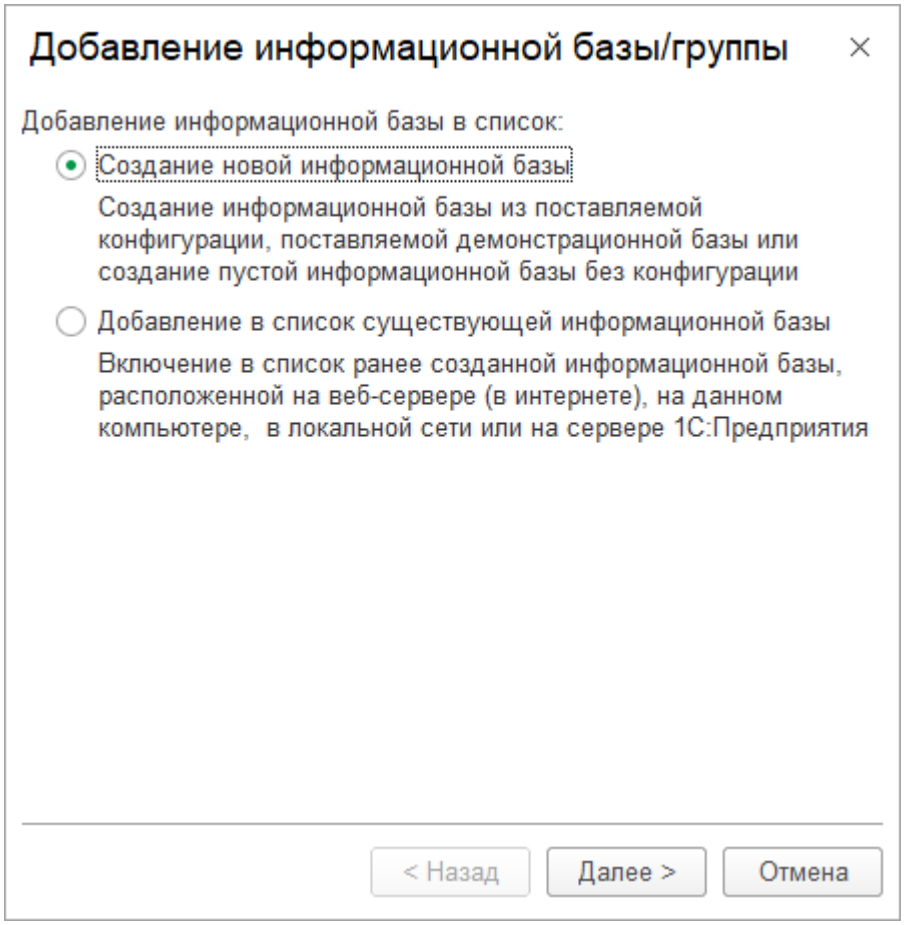


Рисунок 2 – Создание новой информационной базы.

Нажмите кнопку «Далее». На следующем шаге выберите пункт Создание информационной базы без конфигурации... (рис. 3).

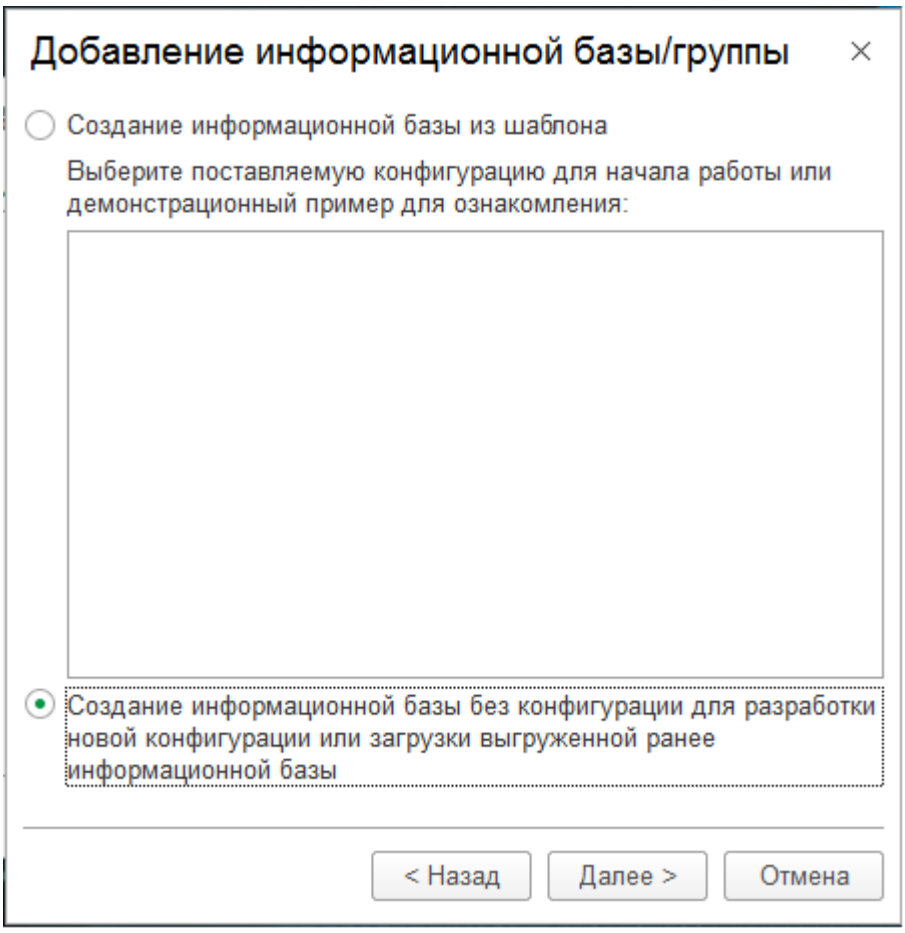


Рисунок 3 – Настройки конфигурации.

Нажмите кнопку «Далее». На следующем шаге задайте наименование вашей информационной базы и выберите тип ее расположения на данном компьютере… (рис. 4).

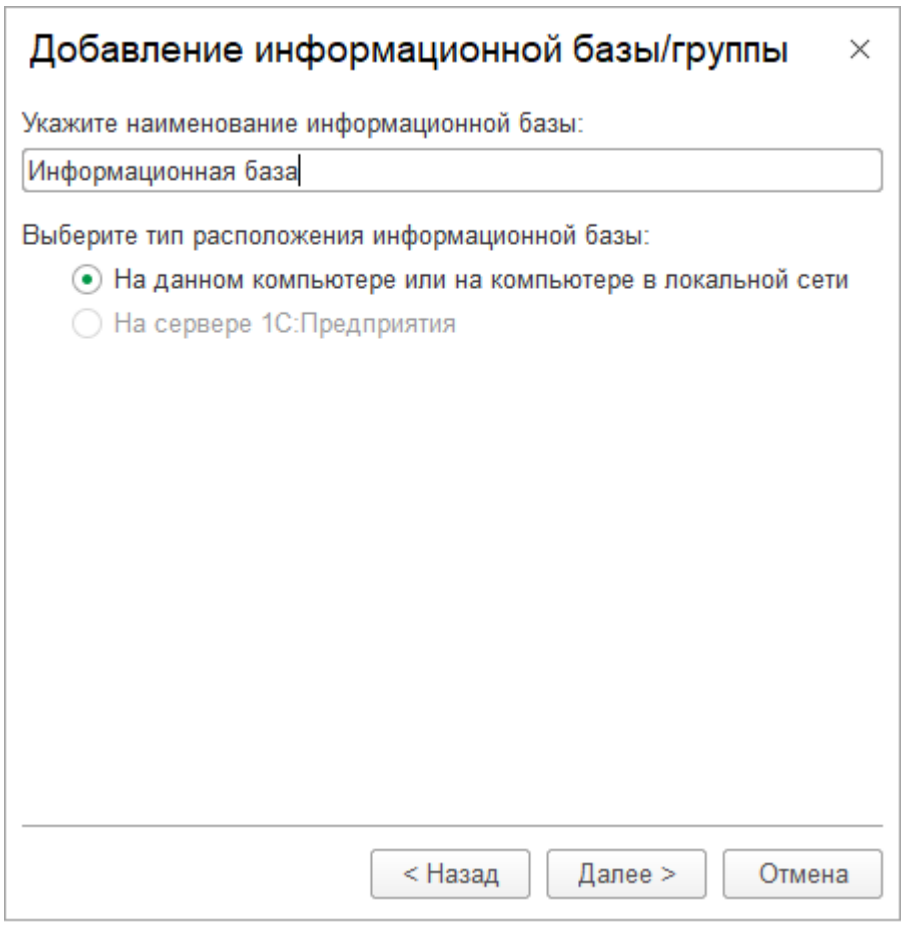


Рисунок 4 – Наименование информационной базы.

Нажмите кнопку «Далее». На следующем шаге укажите каталог для расположения вашей информационной базы. Язык по умолчанию установлен в значение Русский (рис. 5).

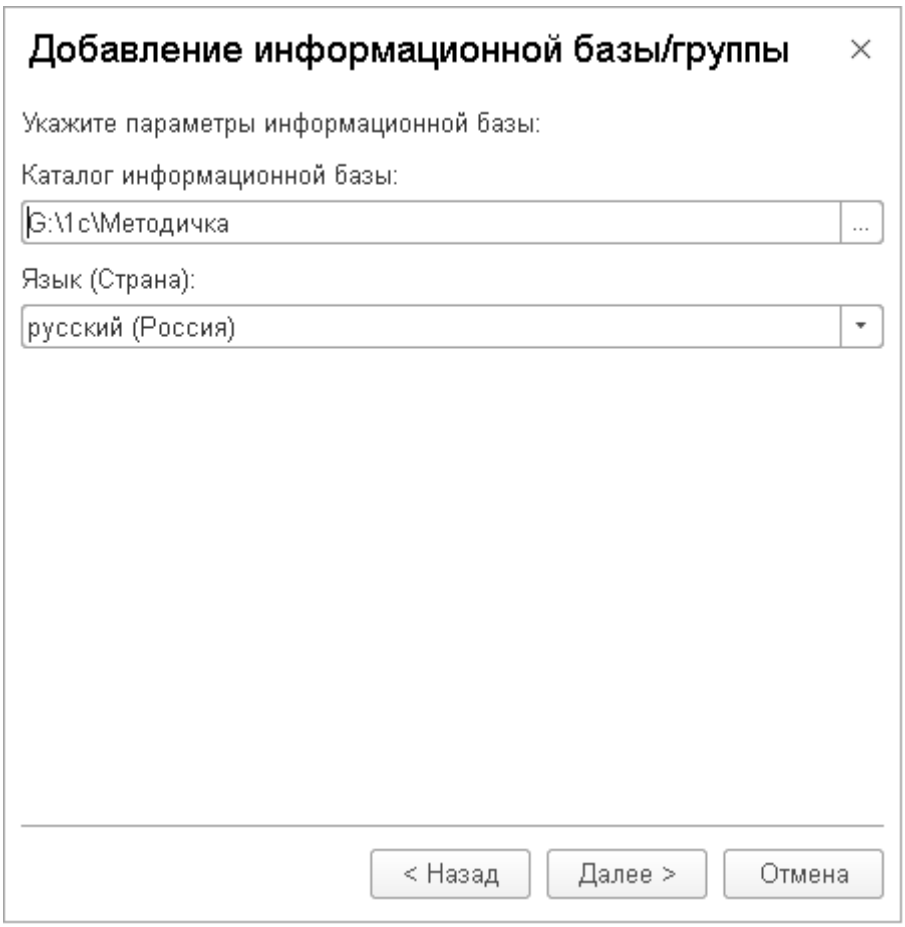


Рисунок 5 – Настройки каталога и языка.

Нажмите кнопку «Далее». На следующем шаге нажмите кнопку Готово (рис. 6).

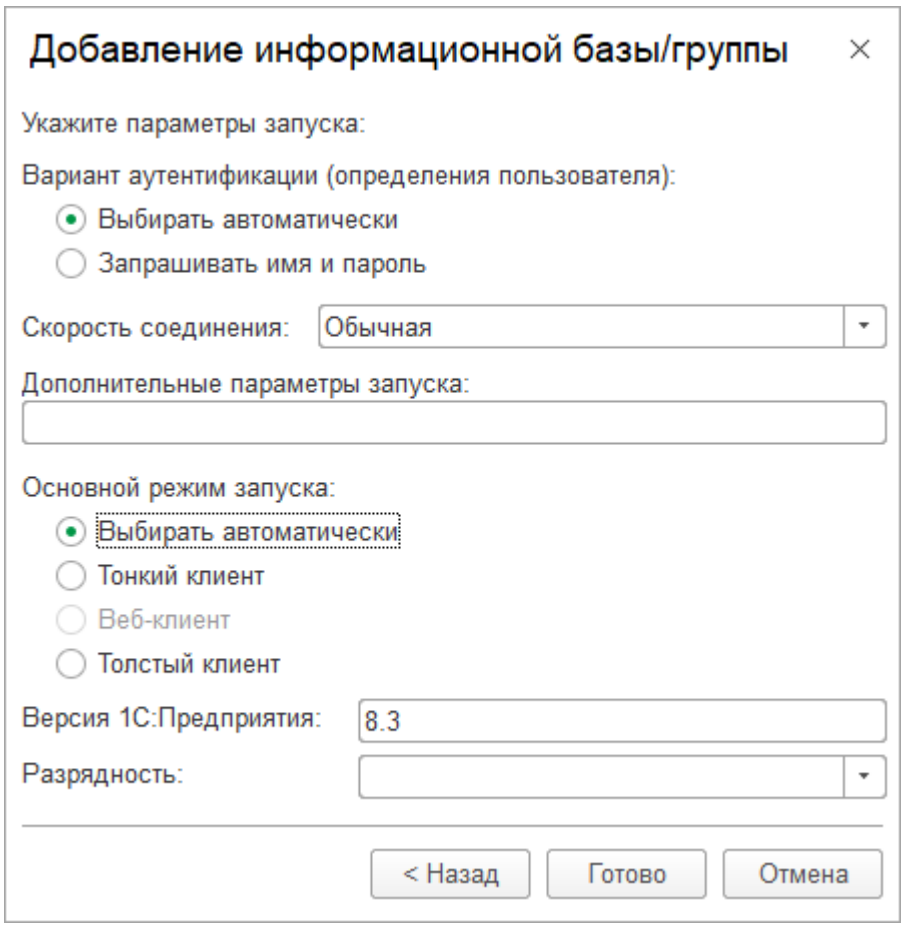


Рисунок 6 – Диалоговое окно с настройками новой базы

В диалоге запуска «1С: Предприятия», в списке информационных баз вы увидите созданную вами новую пустую базу (рис. 7).

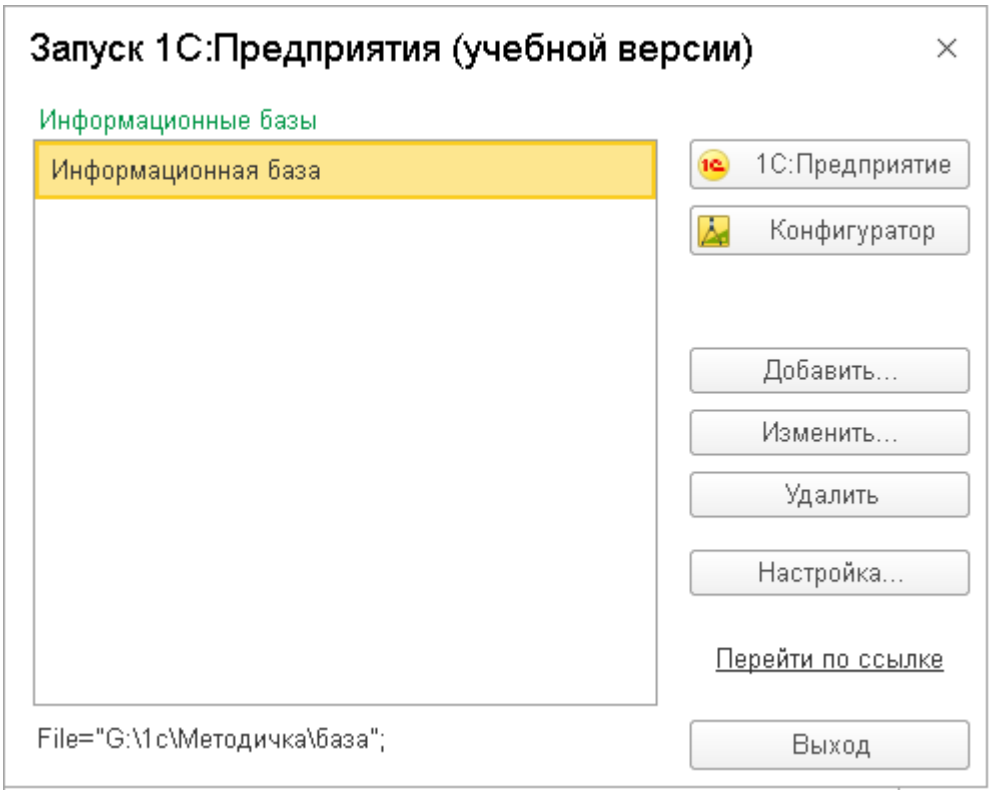


Рисунок 7 – Созданная база.

Конфигуратор

Конфигурация может быть запущена в двух режимах:

- 1С: Предприятие – данный режим используется для запуска и отладки приложения;

- Конфигуратор – используется для разработки и администрирования конфигурации.

Запустим «1С: Предприятие» в режиме Конфигуратор. Для этого нажмем кнопку «Конфигуратор» в диалоге запуска системы. Главное окно конфигуратора показано на рис. 8.

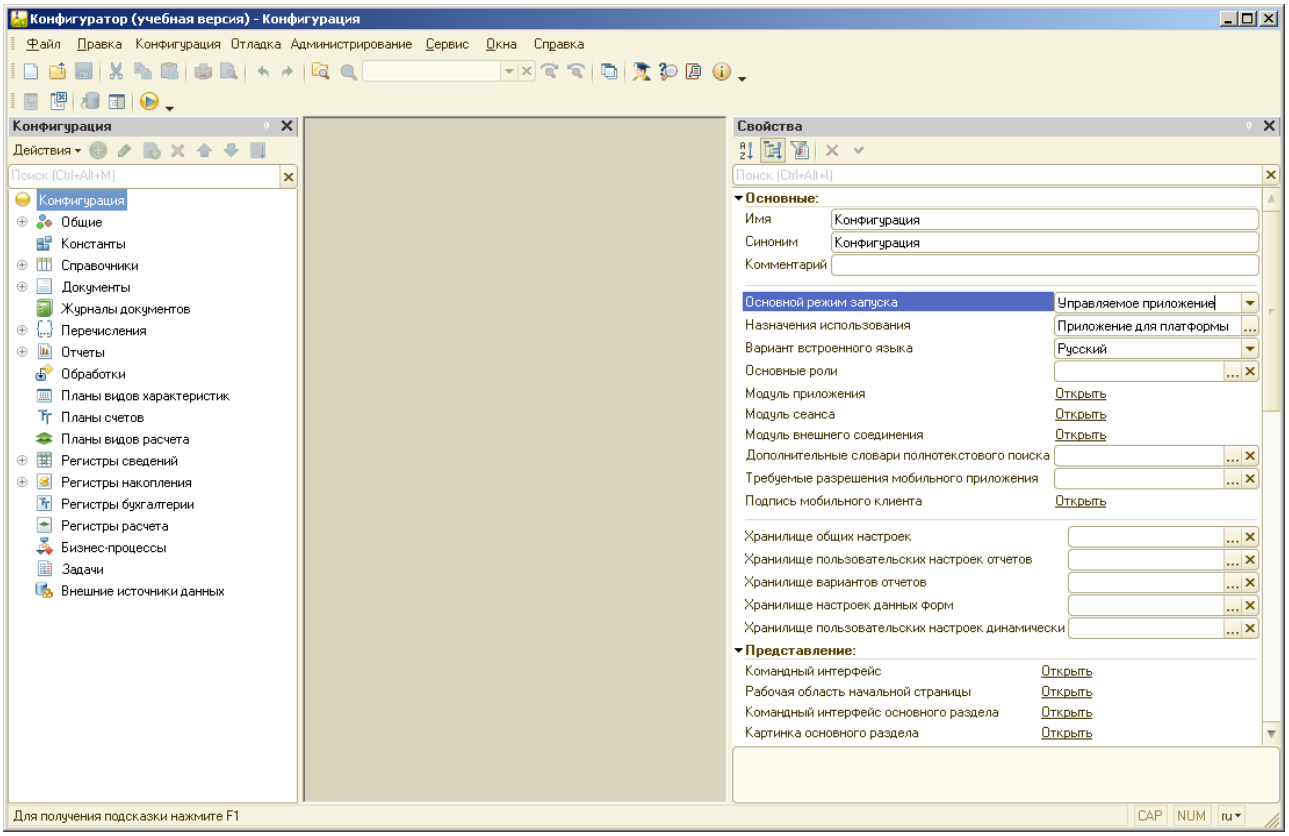


Рисунок 8 – Рабочее окно конфигуратора

Главное меню конфигуратора содержит пункты: Файл, Правка, Конфигурация, Администрирование и т. п. В каждом из этих пунктов содержится много подпунктов, вызов которых обеспечивает выполнение различных действий конфигуратора.

Ниже находится панель инструментов конфигуратора, в которую в виде кнопок-пиктограмм помещены наиболее часто используемые действия, вызываемые из меню.

Слева расположено дерево объектов конфигурации, содержащее в себе практически всю информацию о том, из чего состоит конфигурация. Если дерево не отображается, его можно открыть командой Конфигурация / Открыть конфигурацию.

Справа располагается панель свойств - специальное служебное окно, которое позволяет редактировать все свойства объекта конфигурации и другую связанную с ним информацию. Поскольку разные объекты конфигурации имеют самые разные свойства, содержимое этого окна будет меняться в зависимости от того, какой объект является текущим (на каком объекте конфигурации установлен курсор). Для того чтобы редактировать свойства всей конфигурации, необходимо выделить в дереве корневой узел Конфигурация и дважды щелкнуть на нем.

Рассмотрим наиболее важные свойства конфигурации.

Свойство *Основной режим запуска* позволяет выбрать режим запуска: управляемое приложения или обычное приложение. Начиная с версии 8.2 1С: Предприятие поддерживает **режим управляемого приложения**. Данный режим оптимизирован для использования преимуществ клиент-серверной технологии и значительно облегчает разработку и внедрение приложения. Основные отличия от обычного приложения для разработчиков заключаются в том, что, во-первых, управляемое приложение позволяет задать место исполнения отдельных частей кода (на клиенте или на сервере), а во-вторых, отсутствует необходимость конструирования визуальных форм вручную, поскольку они генерируются автоматически при выборе реквизитов, которые должны отображаться на форме.

Свойство *Назначение использования* позволяет задать тип разрабатываемого приложения: приложение для платформы и/или приложение для мобильной платформы.

Свойства *Модуль приложения*, *Модуль сеанса*, *Модуль внешнего соединения* открывают соответствующие модули.

Свойства *Режим управления блокировкой данных*, *Режим использования модальности*, *Режим совместимости интерфейса* и др. из группы Совместимость отвечают за совместимость с предыдущими версиями платформы.

Объекты конфигурации

На этапе создания конфигурации разработчик анализирует предметную область и требования пользователей, создает объекты конфигурации, настраивает связи между ними путем установки их свойств, визуально конструирует экранные формы и макеты отчетов, описывает поведение различных объектов с помощью конструкций встроенного языка программирования платформы 1С. В результате получается прикладное решение, призванное облегчить работу конечных пользователей.

Структура прикладного решения (конфигурации) определяется составом объектов конфигурации и взаимосвязями между ними. Под **объектами конфигурации** понимаются средства 1С:Предприятия, предназначенные для отражения реальных объектов и явлений предметной области, например, справочник "Контрагенты", документ "Счет" и т.д. Разработчик может создавать объекты, не имеющие явного физического воплощения в предметной области, но необходимые для решения задачи, например, регистры сведений, обработки и т.д.

Объекты конфигурации делятся на общие и прикладные.

Общие объекты конфигурации отображаются в ветви Общие дерева конфигурации (рис.)

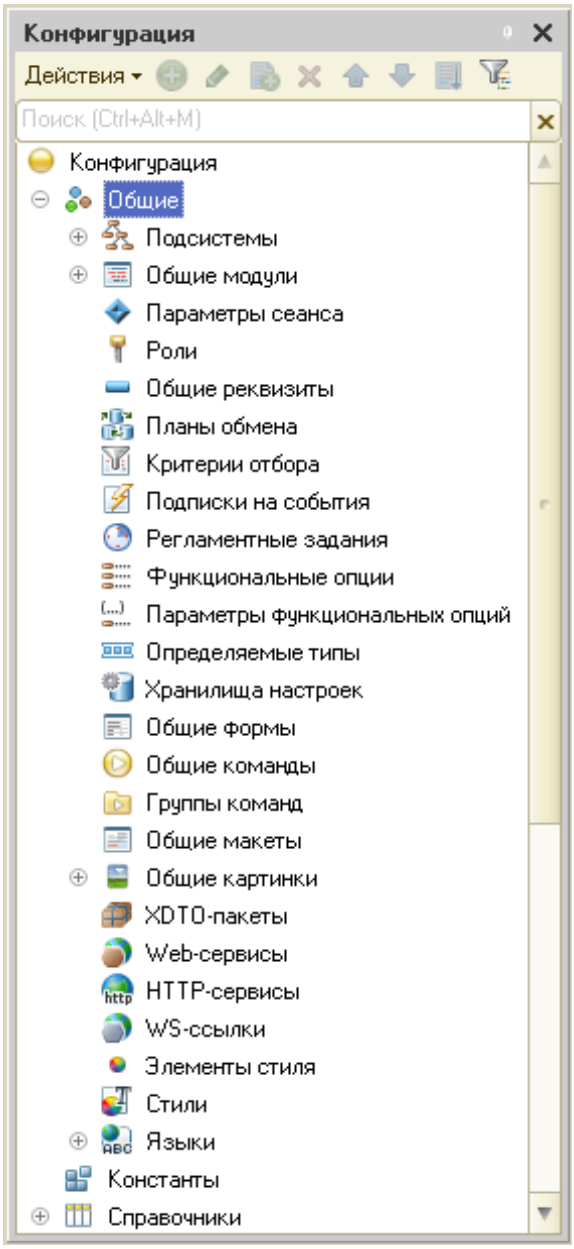


Рис. Дерево конфигурации с раскрытой ветвью Общие

К прикладным объектам относятся все остальные элементы дерева конфигурации, кроме ветви Общие: константы, справочники, документы, журналы документов, перечисления, регистры, отчеты, обработки и др.

Рис. Взаимосвязь объектов конфигурации

Запуск и отладки конфигурации

Теперь проверим наши первые изменения в режиме 1С: Предприятие. Это можно сделать следующими способами:

- нажать на панели инструментов пиктограмму Начать отладку ( );
- выбрать в главном меню команду Отладка / Начать отладку;
- нажать на клавиатуре кнопку F5.

Система анализирует наличие изменений в конфигурации и выдает соответствующий вопрос об обновлении конфигурации базы данных (рис.).

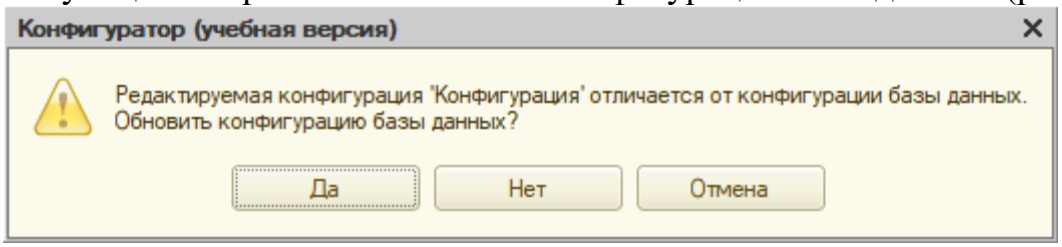


Рис. Вопрос системы об обновлении базы данных

На вопрос конфигуратора ответим Да, и на экране появится окно «1С:Предприятия» (рис.).

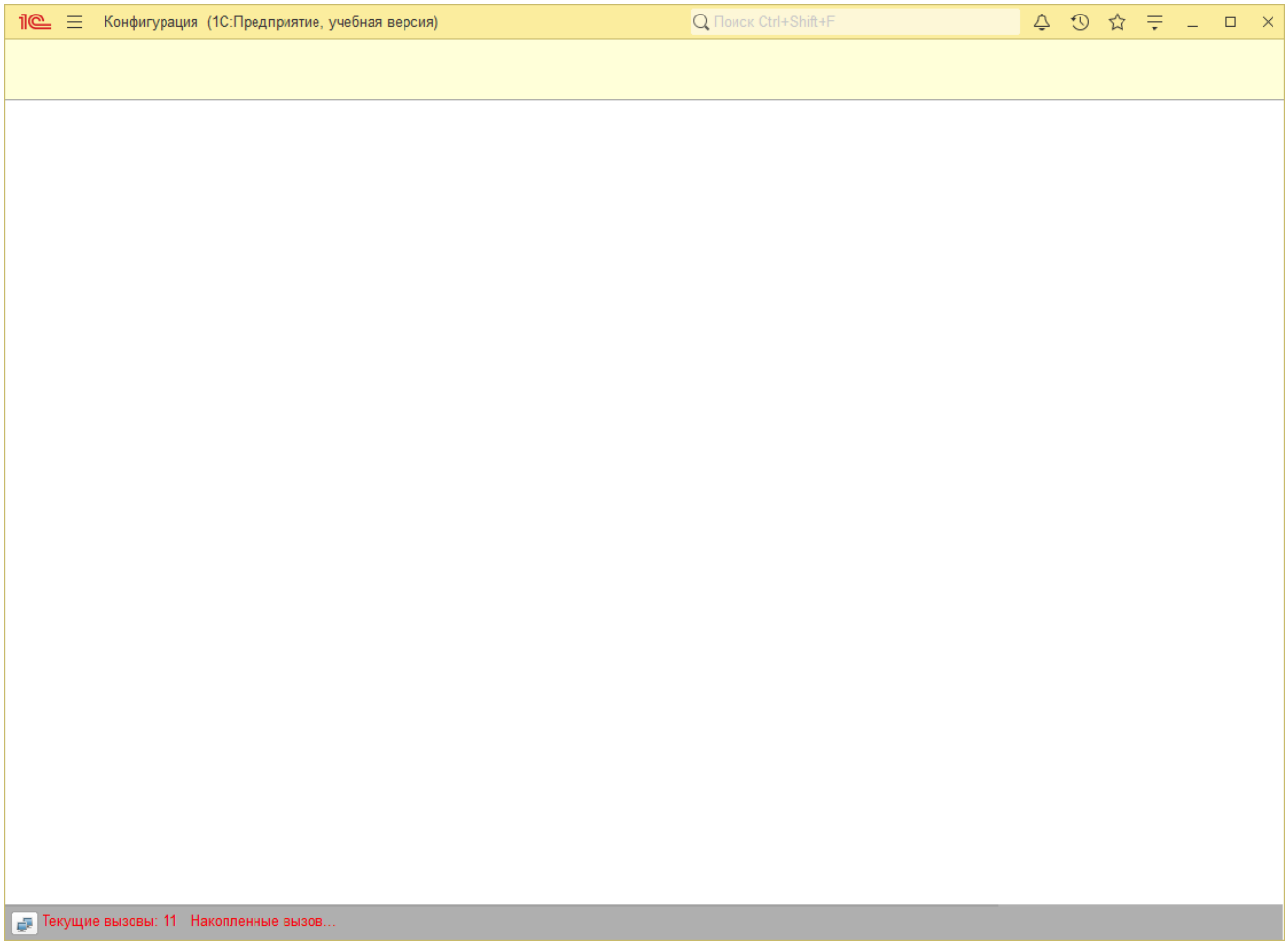


Рисунок 10 – Приложение в режиме отладки.

В заголовке окна мы видим название нашей конфигурации. Пустое пространство – это рабочая область приложения, которая пока ничем не заполнена. Это окно можно просто закрыть.

Основная конфигурация и конфигурация базы данных

Конфигурация, которую создает разработчик, называется *основной конфигурацией*. Пользователи приложения работают с *конфигурацией базы данных*. Вносить изменения в ее структуру можно только путем обновления из основной конфигурации.

Если конфигурация не была создана на основе пустой конфигурации, а путем доработки покупной типовой конфигурации, то существует еще *конфигурация поставщика* от компании, создавшей продукт.

Создание подсистем конфигурации

В начале разработки конфигурации необходимо определить, какие подсистемы она будет содержать.

Чтобы создать новые подсистемы, раскроем ветвь Общие в дереве объектов конфигурации, нажав на «+» слева от нее, и выделим пункт Подсистемы.

Для добавления нового объекта конфигурации можно использовать следующие способы:

1. В командной панели окна конфигурации выбрать команду Действия / Добавить.

2. В командной панели окна конфигурации нажмите кнопку Добавить (с пиктограммой +).

3. В контекстным меню, которое вызывается при нажатии на правую клавишу мыши, выберите пункт Добавить (рис.).

4. Нажать на клавиатуре клавишу Insert (Ins).

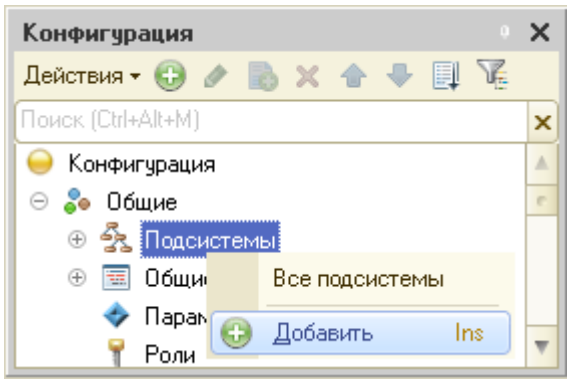


Рис. Добавление подсистемы

После этого откроется окно редактирования объекта конфигурации. Оно предназначено специально для сложных объектов конфигурации и позволяет путем выполнения последовательных действий быстро создавать такие объекты.

Для того чтобы придерживаться правильной последовательности действий, в нижней части окна имеются кнопки Далее и Назад. Кнопка Далее позволяет задавать свойства объекта в нужной последовательности. Кнопка Назад позволяет вернуться на несколько шагов назад, если вы обнаружили, что ранее ввели не все или ошибочные данные. Впоследствии вы сможете задавать свойства объектов, сразу выделяя нужную вам закладку, например, Данные. При открытии окна редактирования объекта конфигурации мы попадаем на закладку Основные (рис.).

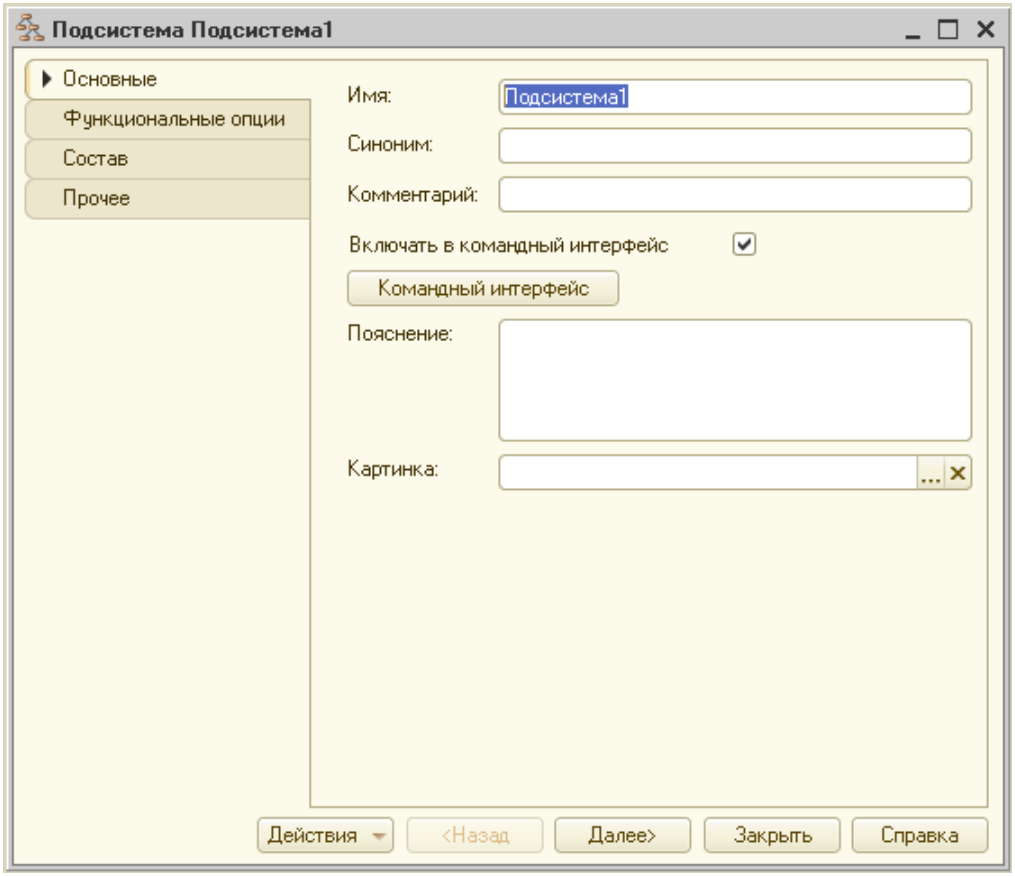


Рис. 2.4. Диалоговое окно создания подсистемы

Зададим имя подсистемы – Бухгалтерия. На основании имени платформа автоматически создаст синоним – Бухгалтерия (рис. 2.4).

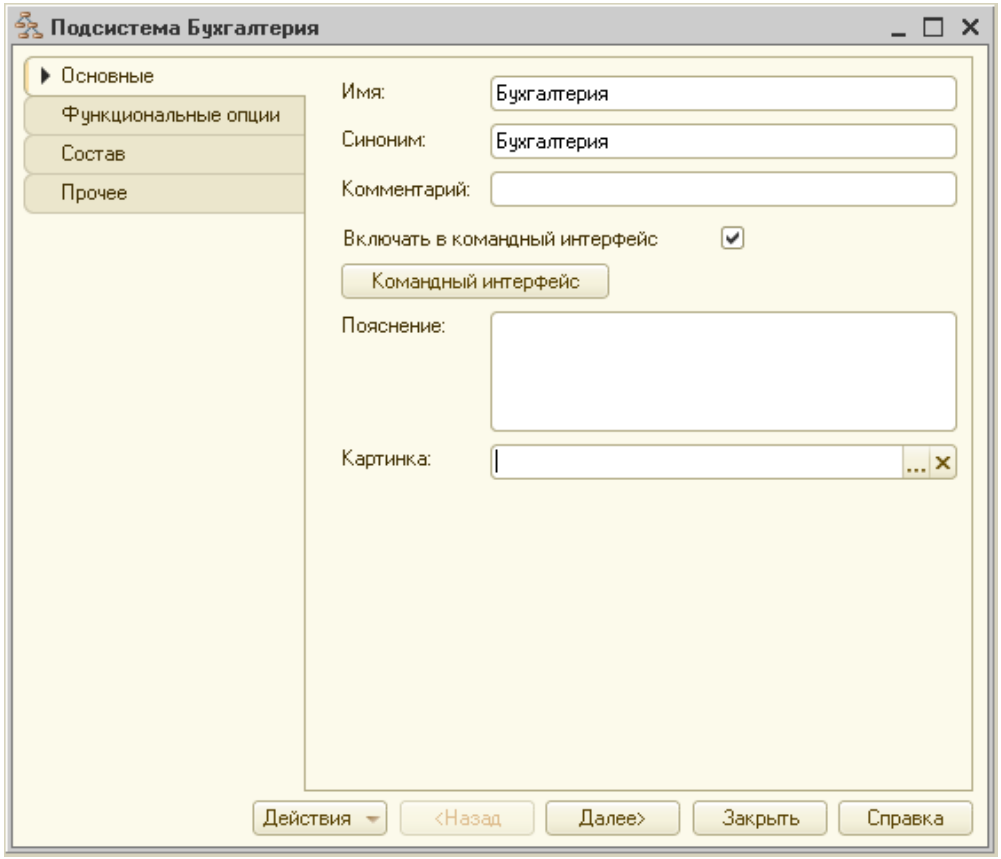


Рис. 2.4. Установка имени и синонима подсистемы

Имя является основным свойством любого объекта конфигурации. При создании нового объекта система автоматически присваивает ему некоторое имя.

Можно использовать имя, присвоенное системой, но лучше заменить его своим, понятным именем. Имя можно задавать любое, главное чтобы оно начиналось с буквы и не содержало некоторых специальных символов (например, пробел).

Для удобства чтения конфигурации принято составлять интуитивно понятные имена и, если они состоят из нескольких слов, удалять пробелы между словами и каждое слово начинать с большой буквы. Имя объекта является уникальным и служит для обращения к свойствам и методам объекта на встроенном языке.

Свойство *Синоним* также есть у любого объекта конфигурации. Оно предназначено для хранения альтернативного наименования объекта конфигурации, которое будет использовано в элементах интерфейса нашей программы, то есть будет показано пользователю. Поэтому на синоним практически нет никаких ограничений, и его можно задавать в привычном для человека виде.

При необходимости можно установить картинку, которая будет отображаться рядом с названием подсистемы. Для этого надо нажать кнопку Картинка и в открывшемся диалоговом окне выбрать картинку (рис.), при этом система позволяет загрузить как новую картинку, так и выбрать стандартную из библиотеки изображений.

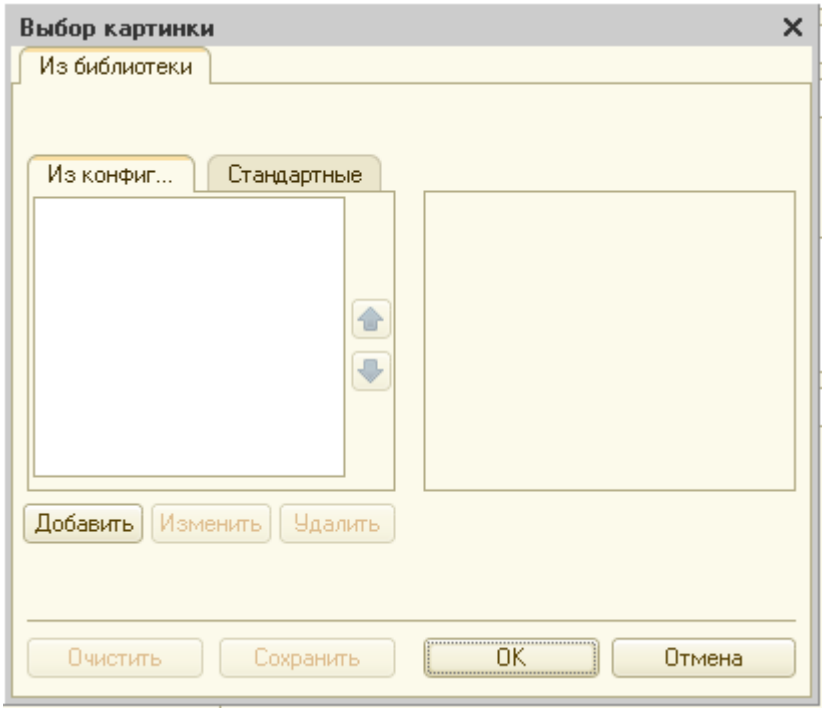


Рис. Диалоговое окно выбора картинки для подсистемы

При нажатии на кнопку Закрыть созданная подсистема появится в дереве конфигурации.

Точно так же добавим подсистемы с именами Закупки, Продажи, РасчетЗарплаты, Предприятие (рис).

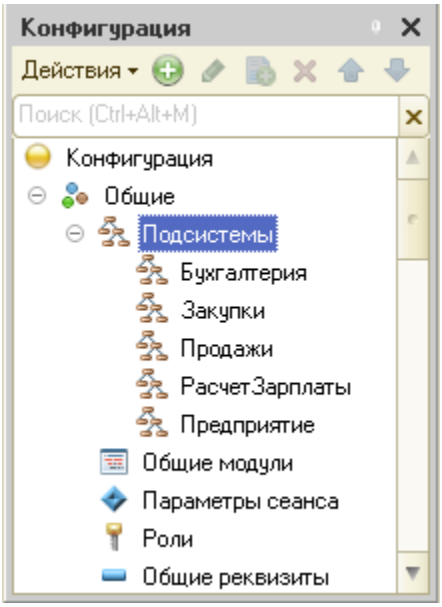


Рис. Список подсистем в дереве объектов конфигурации

При создании остальных объектов конфигурации надо будет указывать, к какой подсистеме они относятся, иначе система будет выдавать ошибки.

Запустим «1С:Предприятие» в режиме отладки. Главное окно приложения со списком подсистем изображено на рис.

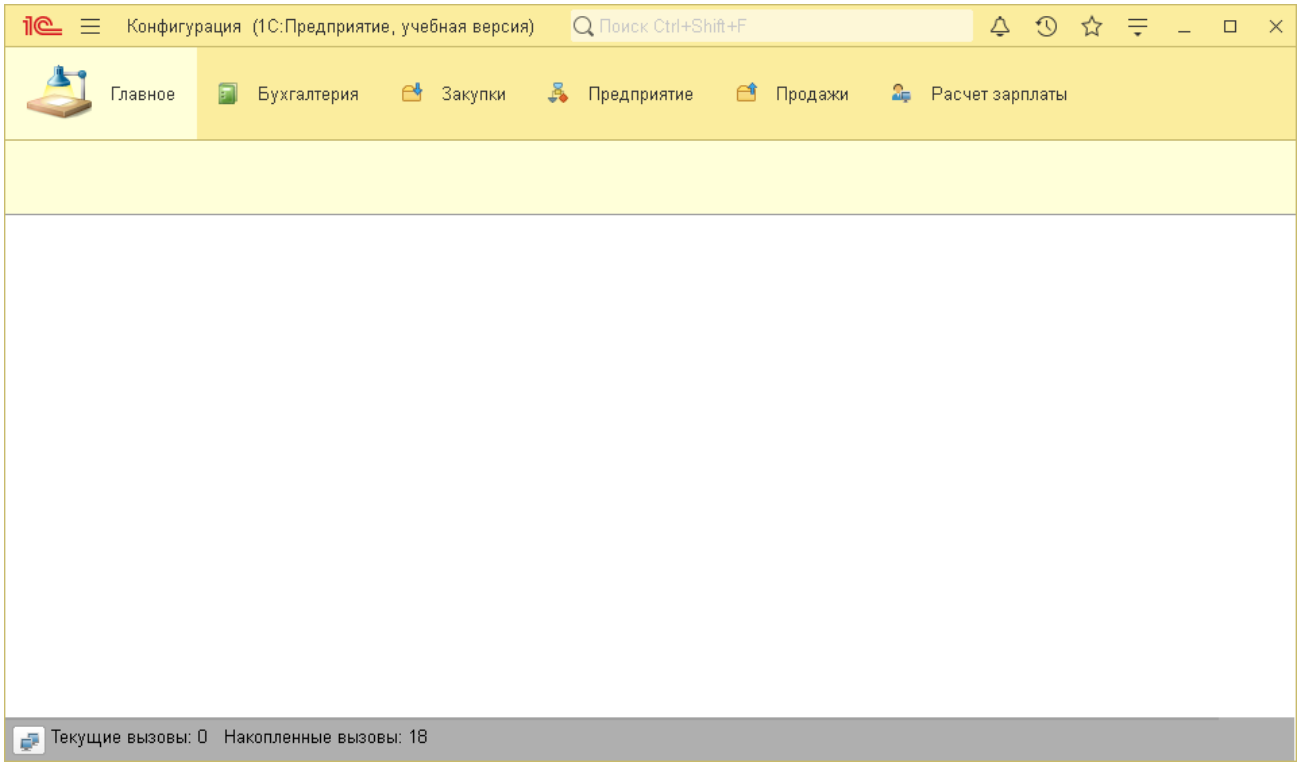


Рис. Основное окно приложения в режиме «1С: Предприятие»

Сразу под заголовком приложения с названием нашей конфигурации и областью системных команд располагается панель разделов приложения, где и отражены созданные нами подсистемы (по умолчанию в алфавитном порядке). Раздел Главное формируется платформой автоматически, всегда располагается первым и предназначен для размещения наиболее часто используемых пользователем объектов конфигурации.

Названия разделов являются гиперссылками, нажав на которые пользователь может открыть связанные с ними документы, справочники, отчеты и т. п. Сейчас состав разделов пуст, так как мы еще не создали наполняющих их объектов конфигурации.

Постановка задачи

Создадим приложение для автоматизации деятельности фирмы, занимающейся продажей канцтоваров. Для этого необходимо хранить информацию о сотрудниках, контрагентах, товарах…

Константы

Константы — это прикладные объекты конфигурации, предназначенные для хранения данных, которые не изменяются или изменяются очень редко. Каждая константа позволяет хранить одно значение.

Например, константы можно использовать для хранения реквизитов организации (наименования, ИНН и др.) или настроек системы. В прикладном решении может быть создано произвольное количество констант.

Создадим константу, содержащую наименование организации. Выделим в дереве объектов элемент Константы и создадим новую константу. В панели свойств установим следующие свойства (рис.):

- Имя – Наименование;
- Тип – Строка;
- Длина – 100.

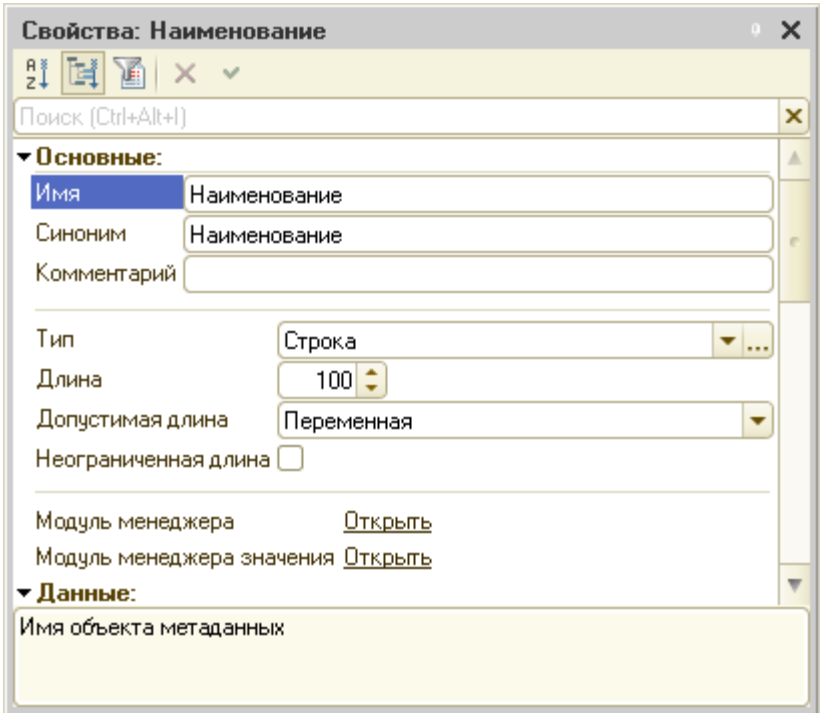


Рис. Панель свойств константы

Чтобы указать, в какой подсистеме будет отображаться константа, необходимо выделить ее в дереве объектов и в контекстном меню выбрать команду Дополнительно. В открывшемся окне дополнительных свойств отметим подсистему Предприятие (рис.).

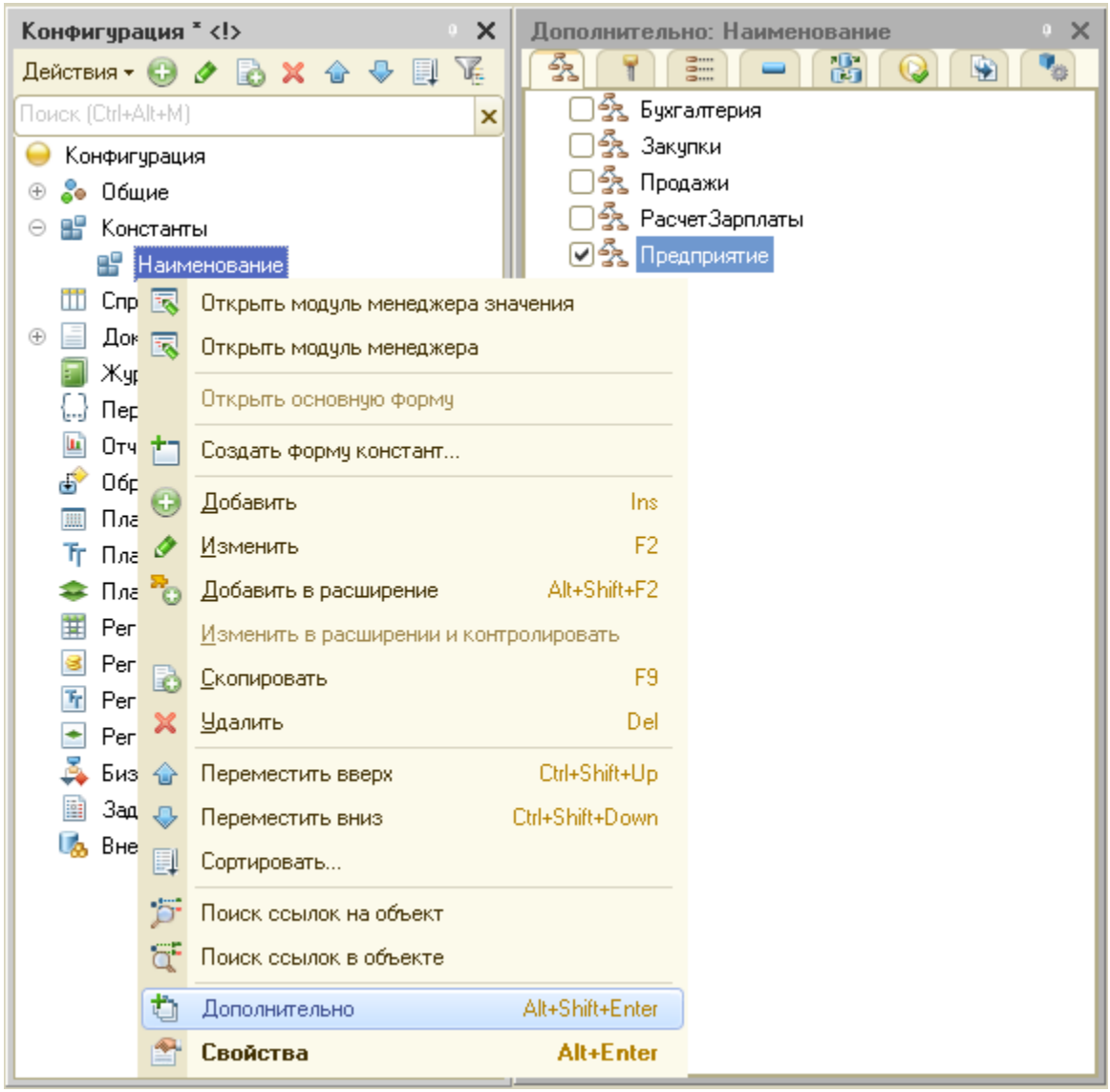


Рис. Выбор подсистемы, в которой будет отображаться константа

Запустим конфигурацию в режиме отладки и ответим утвердительно на вопрос системы об обновлении (рис.).

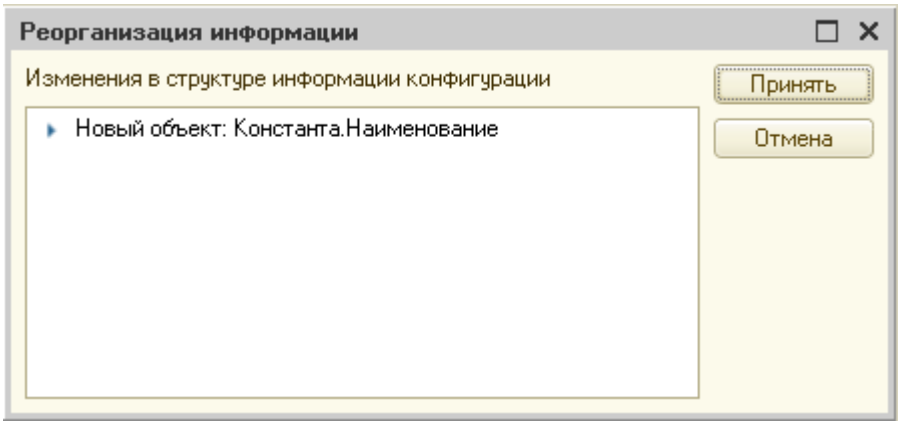


Рис. Предупреждение об обновлении конфигурации

В окне приложения перейдем в раздел Предприятие. Система автоматически создала группу команд Сервис, в которую добавлена команда открытия формы редактирования константы (рис.).

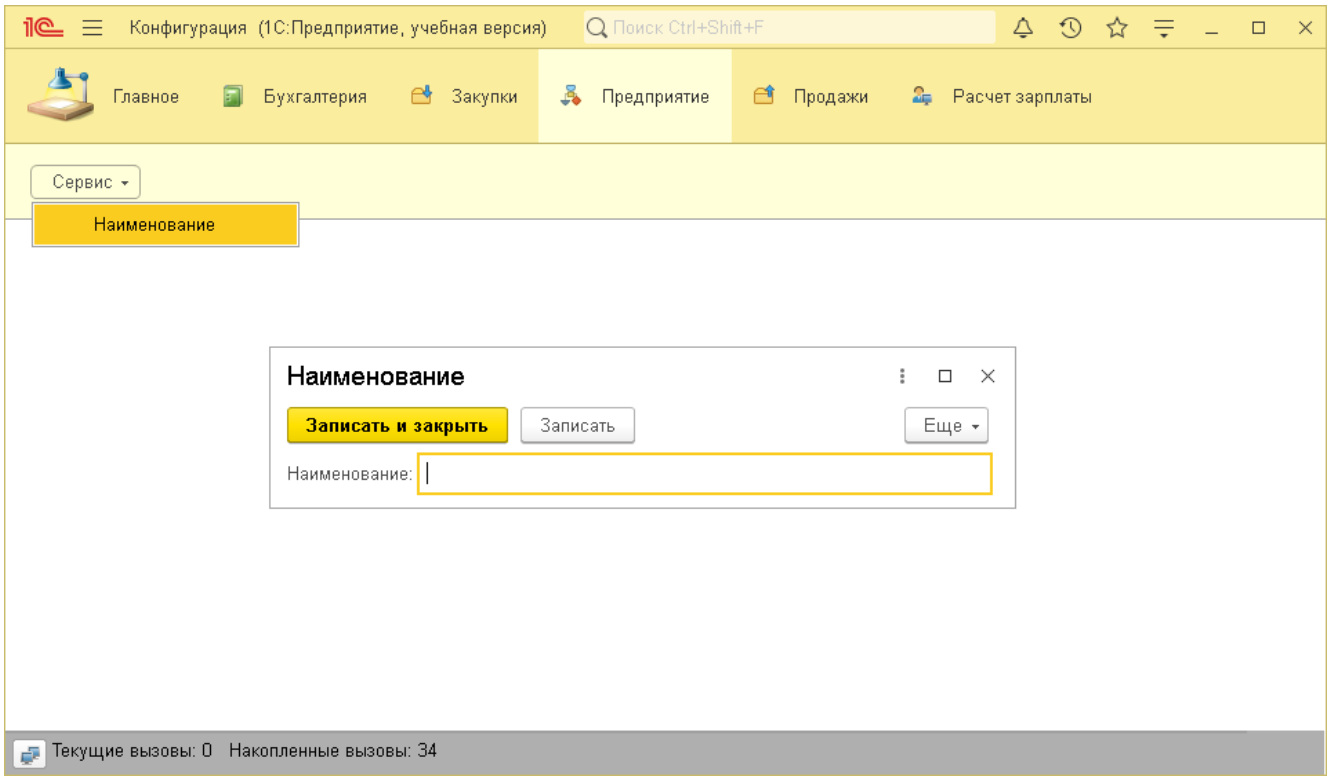


Рис. Форма редактирования константы

Таким же образом создадим следующие константы:

- ПолноеНаименование (многострочный режим – Да), ИНН, РегистрационныйНомерВПФР (синоним – в ПФР), Адрес - тип Строка;
- Дата регистрации - тип Дата, состав даты - Дата.

Проверка заполнения реквизитов объектов конфигурации

Для автоматической проверки корректности ввода данных пользователем в системе можно использовать следующие свойства объекта:

- *Проверка заполнения* – позволяет задавать варианты проверки введенных данных; может принимать значения «Выдавать ошибку» и «Не проверять»; на форме поля, требующие обязательного заполнения, отображаются с красной пунктирной линией внутри;
- *Маска* – представляет собой строку, содержащую набор специальных символов, устанавливающих ограничения для вводимых данных. Символы перечислены в табл.

Табл. Символы маски

№ п/п	Символ маски	Допустимый символ
1.	X	любой символ
2.	!	любой символ, при вводе преобразуется в верхний регистр
3.	N	алфавитно-цифровой символ (буква или цифра)
4.	@	алфавитно-цифровой символ в верхнем регистре, пробел
5.	U	алфавитно-цифровой символ, при вводе преобразуется в верхний регистр
6.	9	Цифра
7.	#	цифра, «-»,«+», пробел
8.	H	шестнадцатеричная цифра

При форматировании строки для разделения символов можно дополнительно использовать некоторые знаки, например «.», «,», «(», «)», «-» и др.

Система 1С позволяет задавать несколько масок для одного и того же объекта. В этом случае они перечисляются через точку с запятой.

Установим проверку заполнения для константы Наименование (рис.).

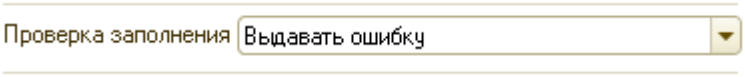


Рис. Фрагмент панели свойств константы Наименование

ИНН () представляет собой 10-значное число, поэтому маска для него будет иметь вид 99999999 (рис.).

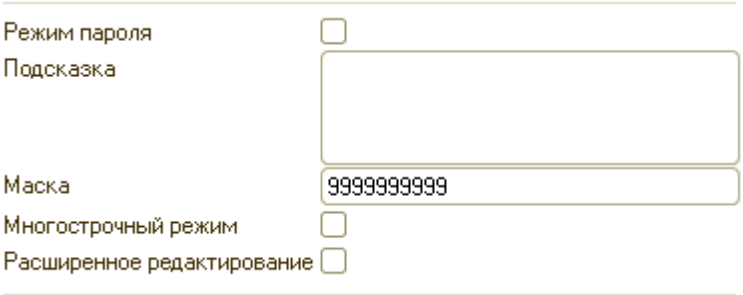


Рис. Фрагмент панели свойств константы ИНН с определением маски ввода

Регистрационный номер в ПФР – это 12-значное число, в котором цифры разделены на группы с помощью знаков «-», например, 123-123-123456. Для корректного хранения данного номера необходимо в панели свойств установить свойство Длина = 14 и задать маску 999-999-999999 (рис.).

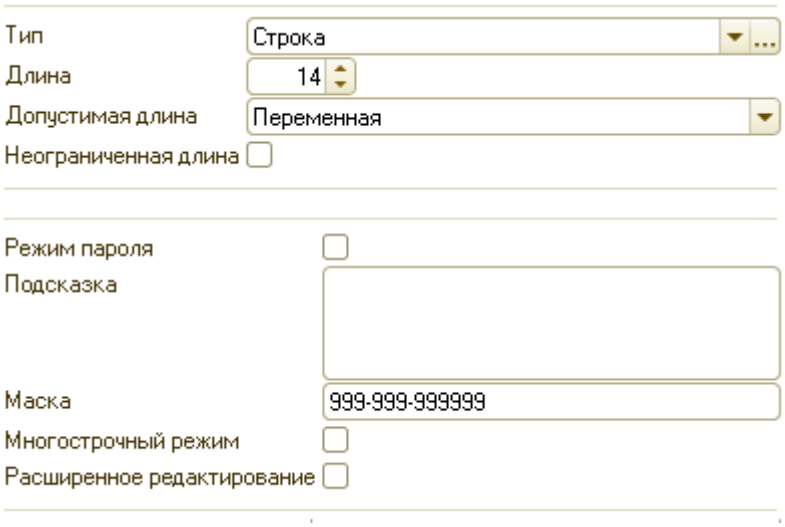


Рис. Фрагменты панели свойств константы РегистрационныйНомерВПФР с определением длины строки и маски ввода

Формы

Формы в системе 1С:Предприятие предназначены для отображения и редактирования информации, содержащейся в базе данных. В режиме управляемого приложения формы также называются *управляемыми*. Их основная особенность заключается в том, что они не создаются вручную из визуальных элементов, а генерируются автоматически на основе выбранных или созданных разработчиком объектов.

До сих пор мы работали только со стандартными формами, которые генерировались системой автоматически при создании объектов конфигурации. Однако разработчик имеет возможность создавать и свои формы.

Формы в системе 1С могут принадлежать конкретным объектам конфигурации, но есть и *общие формы*, существующие отдельно от конкретного объекта конфигурации.

Каждый объект конфигурации может иметь несколько форм, точный список которых зависит от типа объекта. Наиболее часто используются следующие виды форм:

- *форма элемента* используется для создания или редактирования элемента объекта конфигурации;
- *форма группы* очень похожа на форму элемента, но предназначена для создания группы;
- *форма списка* используется для отображения списков элементов объектов конфигурации;
- *форма выбора* предназначена для выбора одного или нескольких элементов объектов конфигурации.

Пока все созданные нами константы отображались в разных автоматически генерируемых системой формах.

Создадим общую форму, содержащую сведения об организации. Для этого в дереве конфигурации раскроем ветвь Общие и найдем объект Общие формы. На

первой странице конструктора общих форм выберем тип Форма констант и зададим имя и синоним формы (рис.).

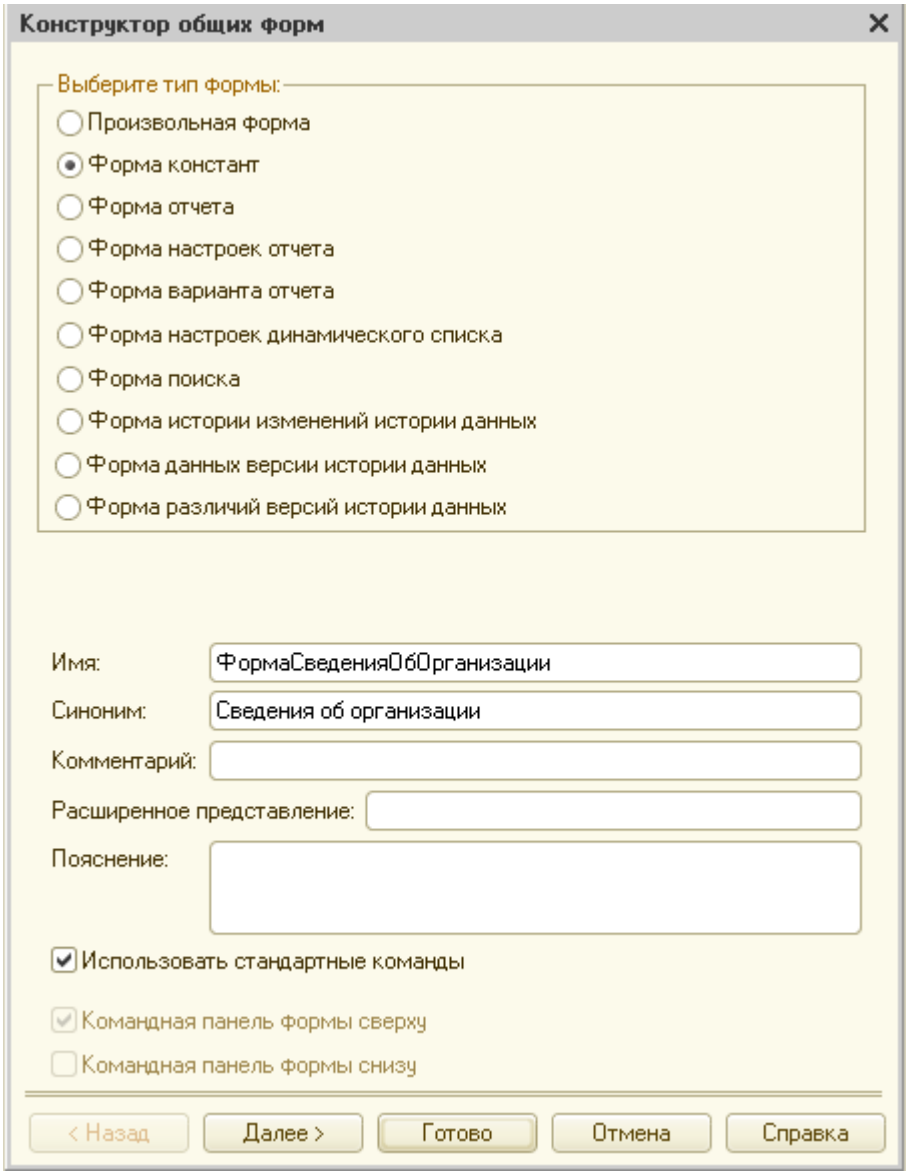


Рис. Конструктор общих форм

Нажмем кнопку Далее. На следующей странице конструктора можно выбрать константы, отображаемые на форме. Нам нужны все константы, поэтому нажмем кнопку Готово (рис.).

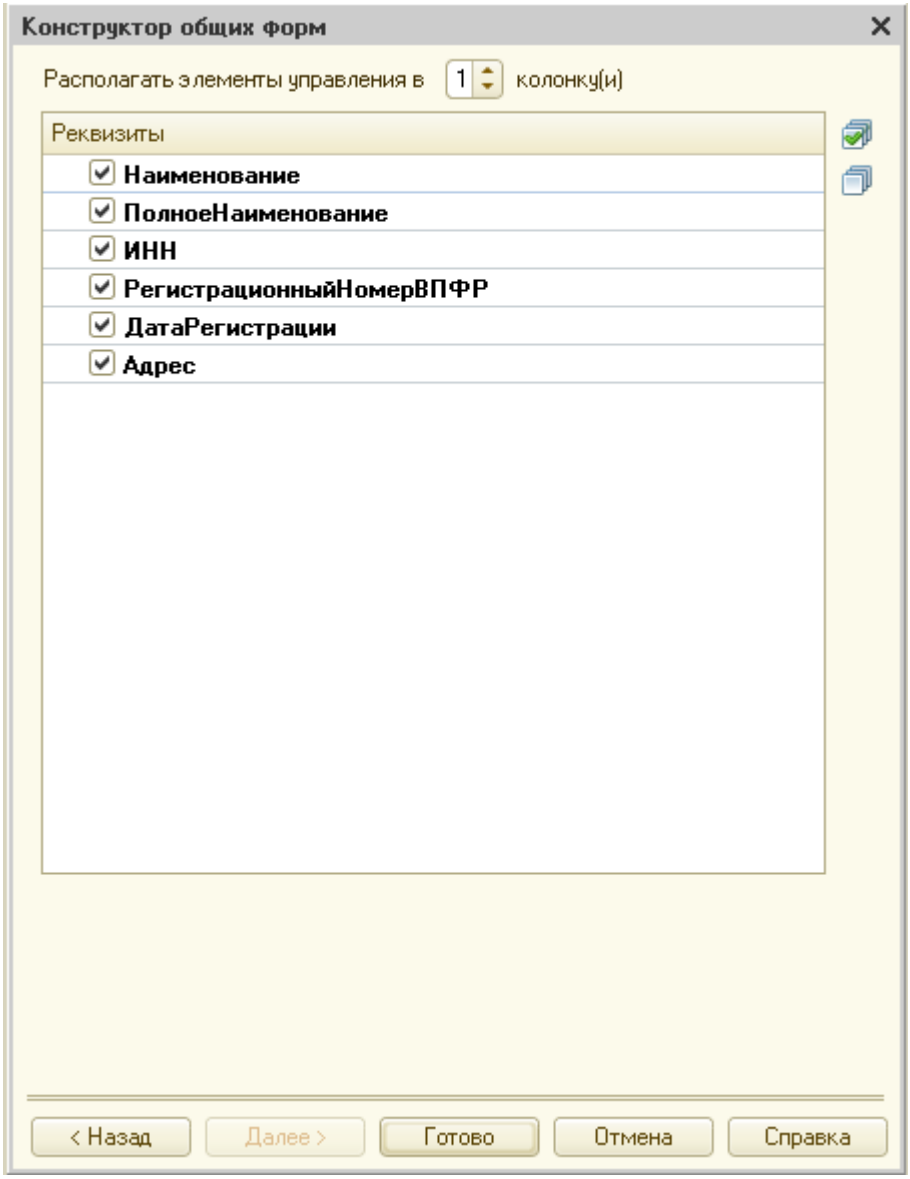


Рис. Выбор констант для отображения на форме

Конструктор форм открывается только при создании формы. В дальнейшем для редактирования свойств формы и состава элементов надо будет использовать панель свойств и редактор формы.

Чтобы система не выдавала ошибку, для формы также необходимо определить принадлежность к подсистеме Предприятие на панели свойств Дополнительно.

После этого откроется окно редактирования формы (рис.).

В верхней левой части окна редактирования формы содержится список элементов, которые будут размещены на форме. Он автоматически заполняется реквизитами объекта конфигурации.

В правой части отображается список реквизитов формы. Один из реквизитов может быть назначен *основным* (выделяется жирным), тогда поведение формы будет определяться именно им.

В нижней части показана форма, автоматически генерируемая на основании составленного разработчиком списка элементов.

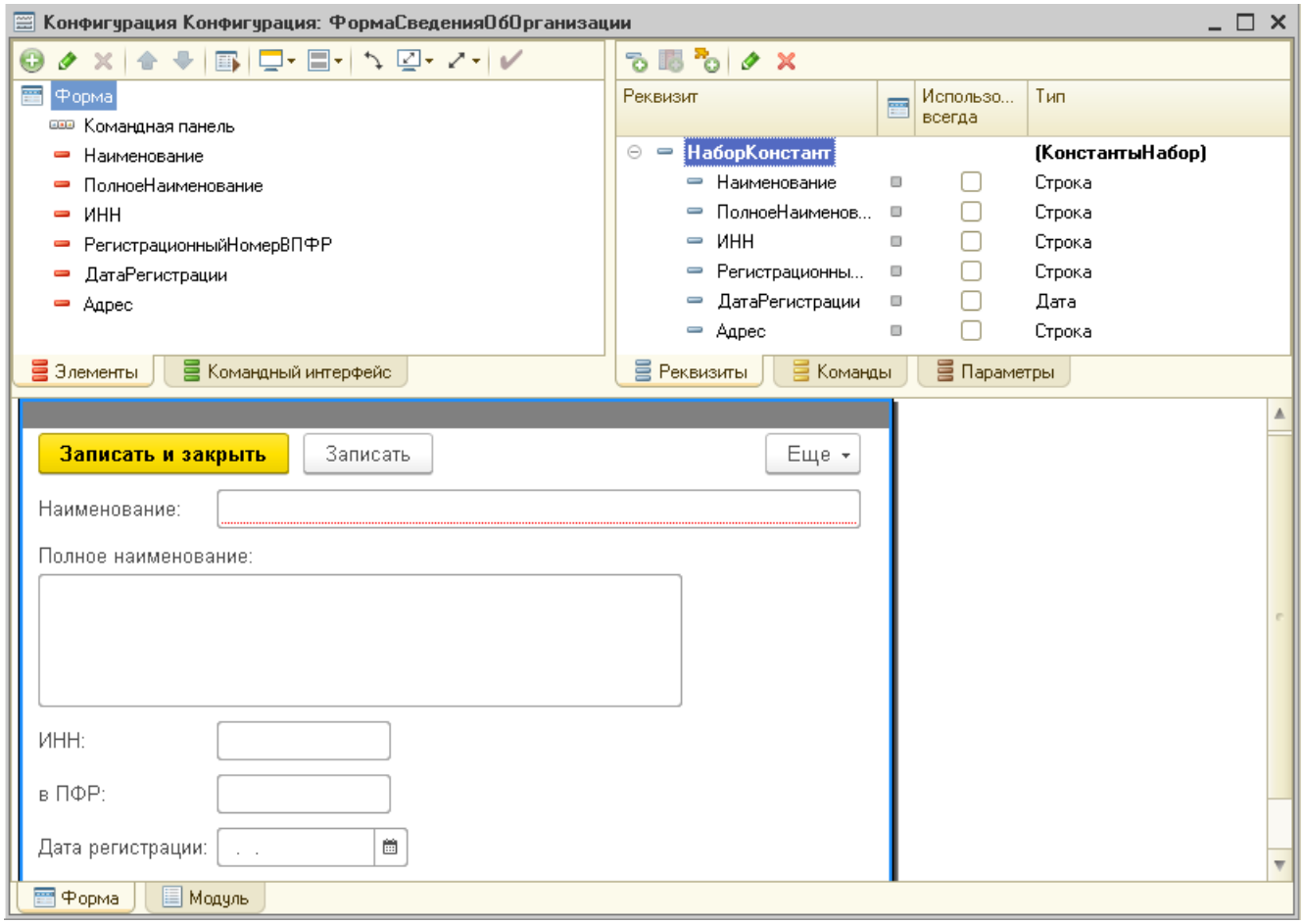


Рис. Окно редактирования формы

Для формы констант основным реквизитом является *набор констант*, выбранных для отображения на форме.

Разработчик может влиять на содержимое формы, определяя состав и порядок элементов, а также определенным образом группируя элементы формы с помощью таких элементов, как страницы, группы, таблицы и т.д.

Расположим общие сведения об организации, коды и регистрационные номера и контактные данные на разных вкладках. Для этого нажмем кнопку Добавить ( ) над списком элементов формы. Откроется окно выбора типа элемента формы (рис.).

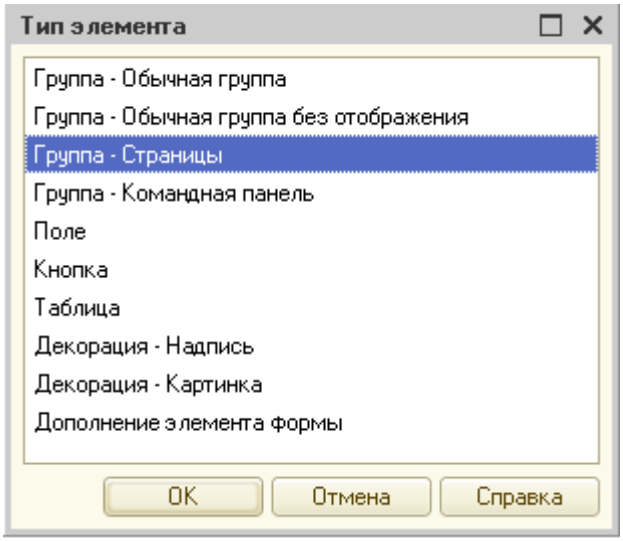


Рис. Окно выбора типа элемента формы

Выберем элемент Группа – Страницы и нажмем кнопку ОК. В списке элементов формы появится группа Группа1, выделим ее и добавим в нее три элемента типа Группа – Страница:

- Группа2 – заголовок - Общие сведения;
- Группа3 – заголовок - Коды и регистрационные номера, группировка – Горизонтальная, если возможно;
- Группа4 – заголовок - Контакты.

Распределим элементы между группами согласно рис.

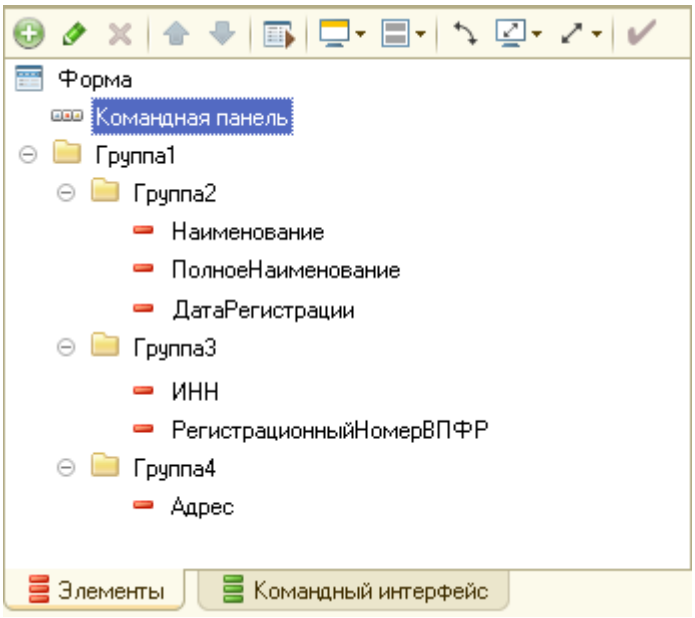


Рис. Окно элементов формы после создания группы Страницы

Далее сгруппируем элементы, добавив следующие элементы:

- Группа5 – тип Группа – Обычная группа, заголовок – Дополнительно, группировка – Горизонтальная, если возможно, поведение – Свертываемая;
- Группа6 – тип Группа – Обычная группа, заголовок – Коды, группировка – Вертикальная, Ширина 25;

- Группа7 – тип Группа – Обычная группа, заголовок – Коды, группировка – Вертикальная.

Перетащим в новые группы элементы согласно рис.

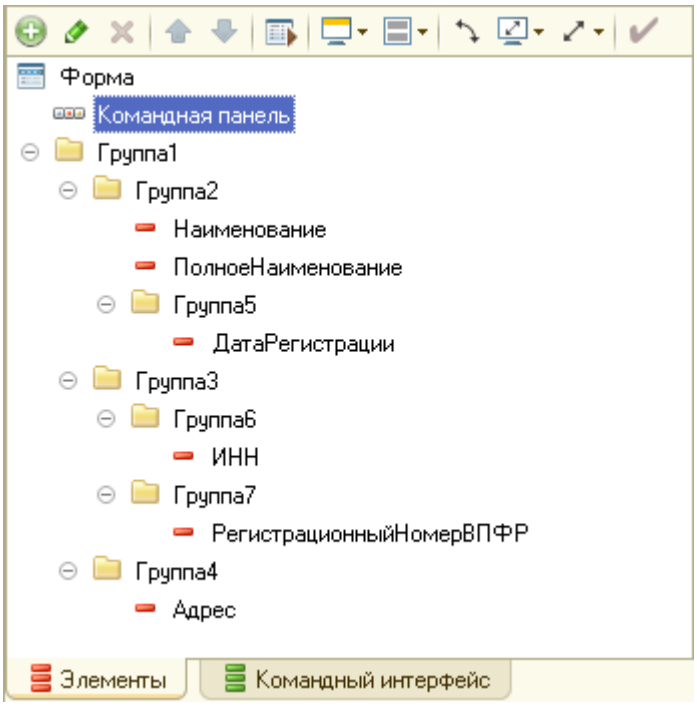


Рис. Окно элементов формы после группировки

Каждый элемент на форме имеет набор свойств, определяющих его вид, расположение, оформление и т.д. Чтобы открыть панель свойств элемента формы, надо дважды щелкнуть по нему или вызвать команду Свойства из контекстного меню элемента.

Для поля ПолноеНаименование откроем панель свойств и в списке ПоложениеЗаголовка выберем пункт Лево (рис.).

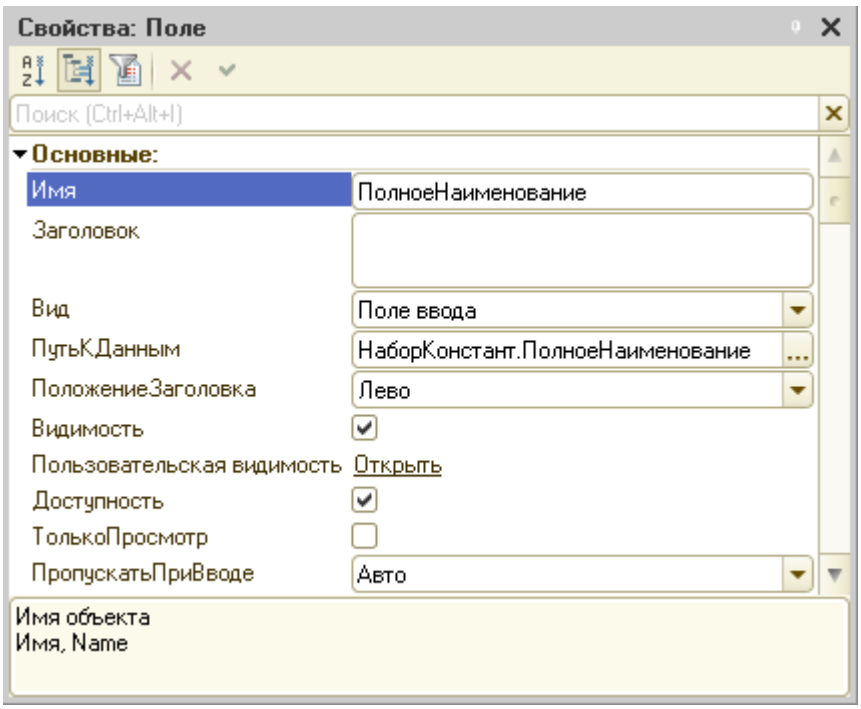


Рис. Панель свойств поля формы ПолноеНаименование

Запустим приложение в режиме отладки и откроем форму Сведения об организации в меню Сервис. Вкладки Основные сведения и Коды и регистрационные номера после редактирования показаны на рис.

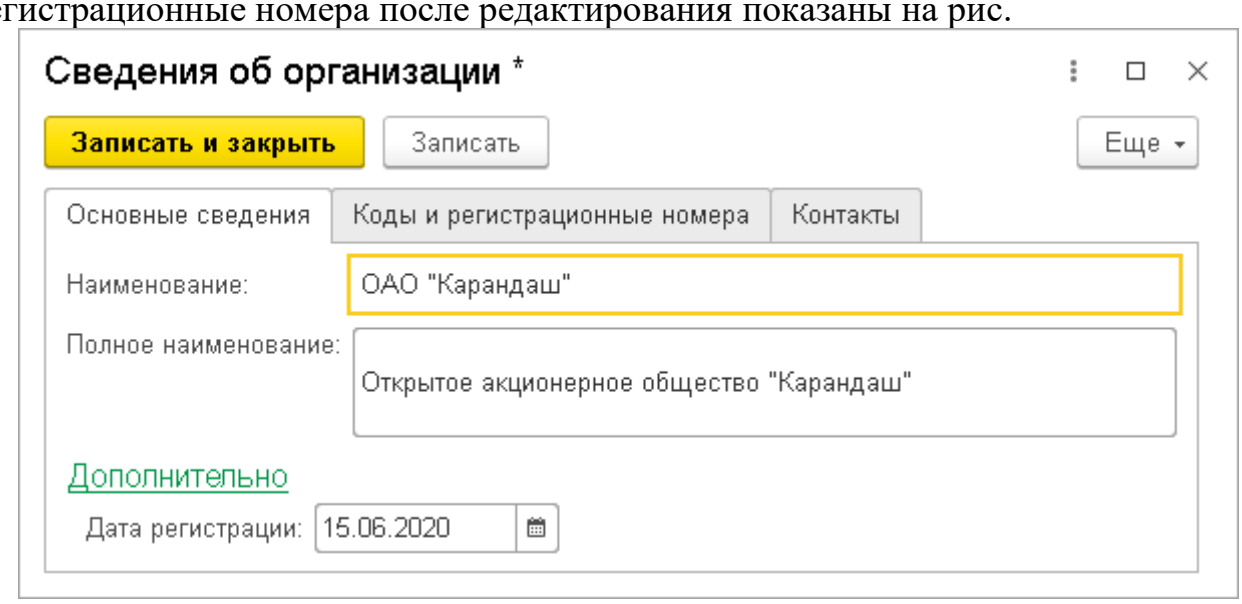


Рис. Вкладка Основные сведения формы констант

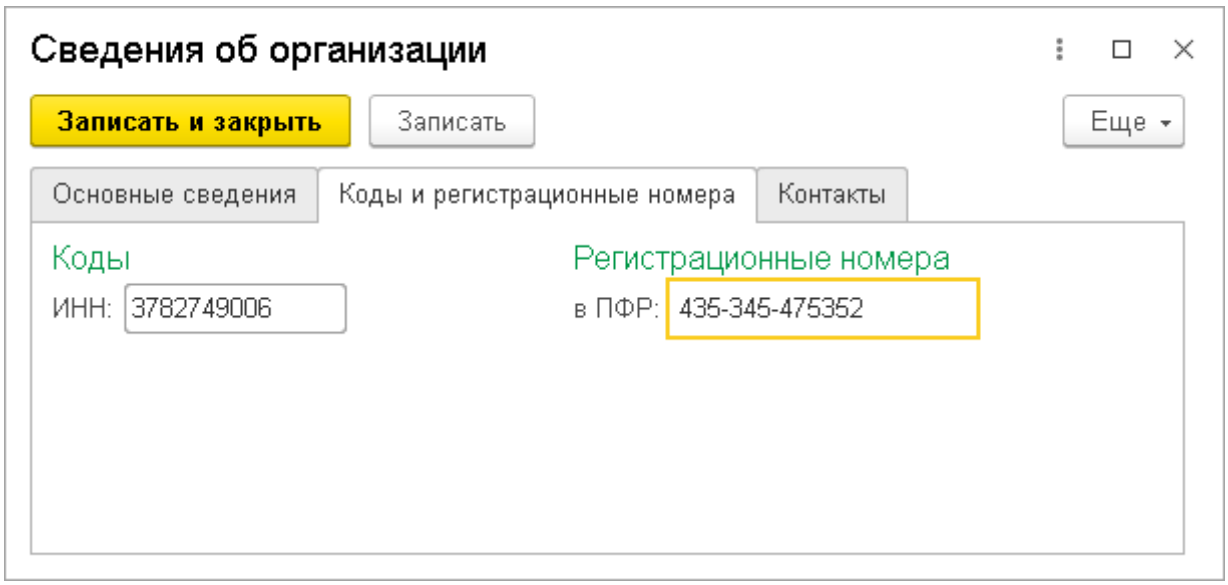


Рис. Вкладка Коды и регистрационные номера формы констант

Для сохранения введенных данных необходимо нажать кнопку Записать и закрыть.

Перечисления

Перечисление – это прикладной объект конфигурации, который используется для хранения списка предопределенных значений. Список значений перечисления задаются разработчиком и пользователь изменить их не может. Традиционно перечисление используется как список выбора для таких реквизитов, как пол, день недели, месяц и т.д.

Создадим новое перечисление Тип организации. На вкладке Подсистемы отметим подсистему Предприятие. На вкладке Данные зададим значения ЮрЛицо (синоним Юр. лицо) и ФизЛицо (синоним Физ. лицо) (рис.).

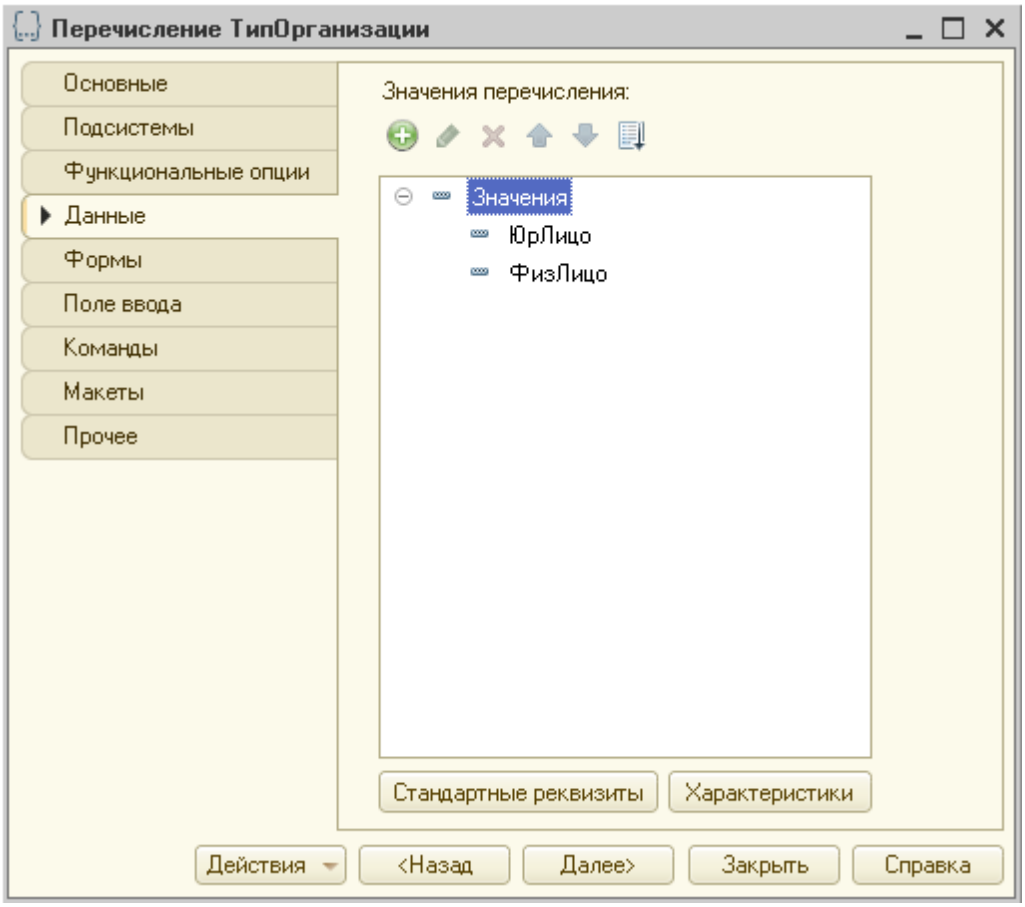


Рис. Задание списка значений перечисления Тип организации

Создадим константу ТипОрганизации (тип ПеречислениеСсылка.ТипОрганизации) и добавим ее в подсистему Предприятие.

Созданная константа автоматически добавляется в Набор констант, но чтобы она отобразилась на форме констант, ее надо вручную перетащить в дерево элементов формы в группу Группа5 (рис.)

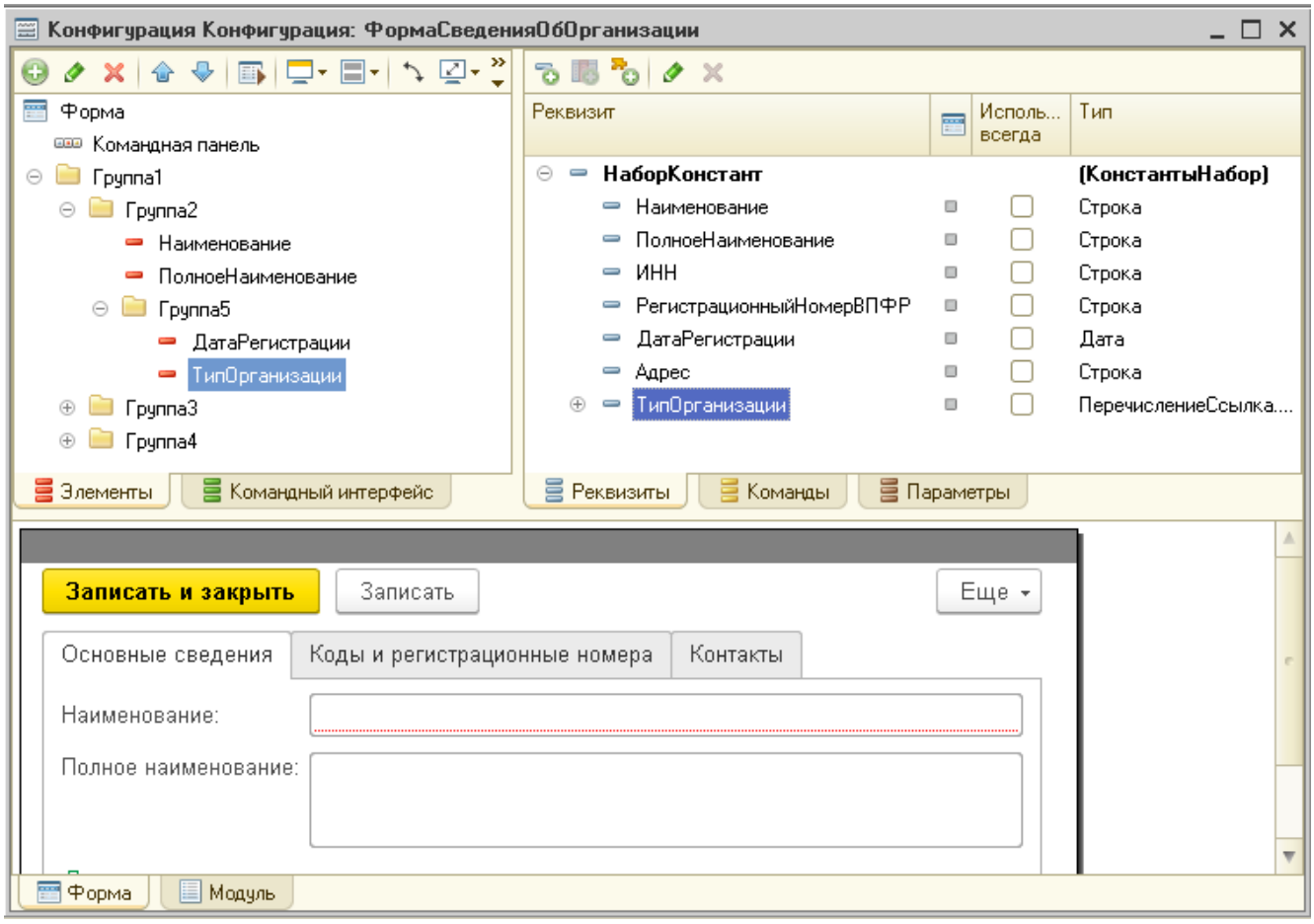


Рис. Добавление новой константы из набора НаборКонстант на форму

Запустим приложение в режиме отладки и выберем значение для типа организации (рис.).

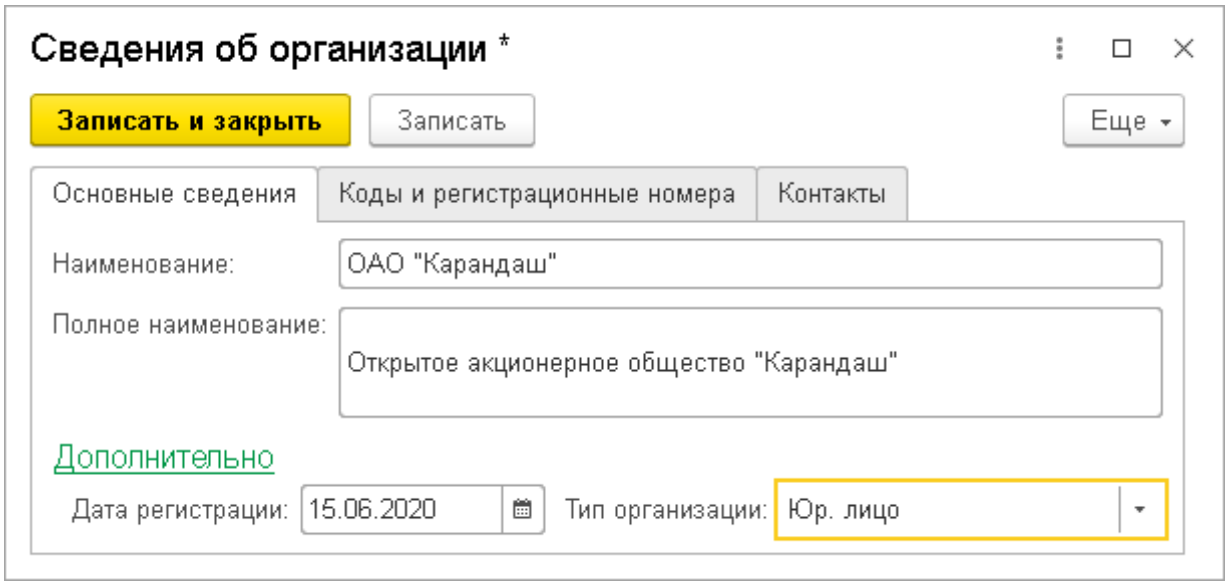


Рис. Форма констант после добавления константы ТипОрганизации

Настройка командного интерфейса в конфигураторе

Командный интерфейс 1С – это часть пользовательского интерфейса, отвечающая за удобство навигации по прикладному решению. Командный интерфейс является управляемым, причем настраивать его может как разработчик в режиме конфигуратора, так и пользователь в режиме отладки.

Конфигуратор предоставляет разработчику следующие редакторы, предназначенные для настройки различных элементов интерфейса приложения:

- *редактор командного интерфейса конфигурации* предназначен для настройки панели разделов и позволяет установить порядок разделов (подсистем) и их видимость для различных групп пользователей;
- *редактор командного интерфейса основного раздела* предназначен для настройки команд начальной страницы (раздела Главное);
- *редактор командного интерфейса* позволяет настроить состав и видимость команд на панели команд подсистемы;
- *редактор Все подсистемы* позволяет редактировать состав и порядок разделов, а также настроить панель команд для каждой подсистемы;
- *редактор интерфейса клиентского приложения* позволяет настраивать расположение стандартных панелей в основном окне приложения.

Все редакторы командного интерфейса можно вызвать из контекстного меню различных элементов дерева объектов конфигурации. Способы вызова редакторов перечислены в табл.

№	Редактор	Доступ к редактору
1	Редактор командного интерфейса конфигурации	выделить корневой узел дерева с наименованием конфигурации и в контекстном меню выбрать команду Открыть командный интерфейс конфигурации
2	Редактор командного интерфейса основного раздела	выделить корневой узел и в контекстном меню выбрать команду Открыть командный интерфейс основного раздела
3	Редактор командного интерфейса	открыть ветвь Общие / Подсистемы, выделить нужную подсистему и в контекстном меню выбрать команду Открыть командный интерфейс
4	Редактор Все подсистемы	- выделить корневой узел и в контекстном меню выбрать команду Все подсистемы; - открыть ветвь Общие, выделить элемент Подсистемы и в контекстном меню выбрать команду Все подсистемы
5	Редактор интерфейса клиентского приложения	выделить корневой узел и в контекстном меню выбрать команду Открыть интерфейс клиентского приложения

Как видно из табл., большая часть редакторов доступна из контекстного меню конфигурации (рис.).

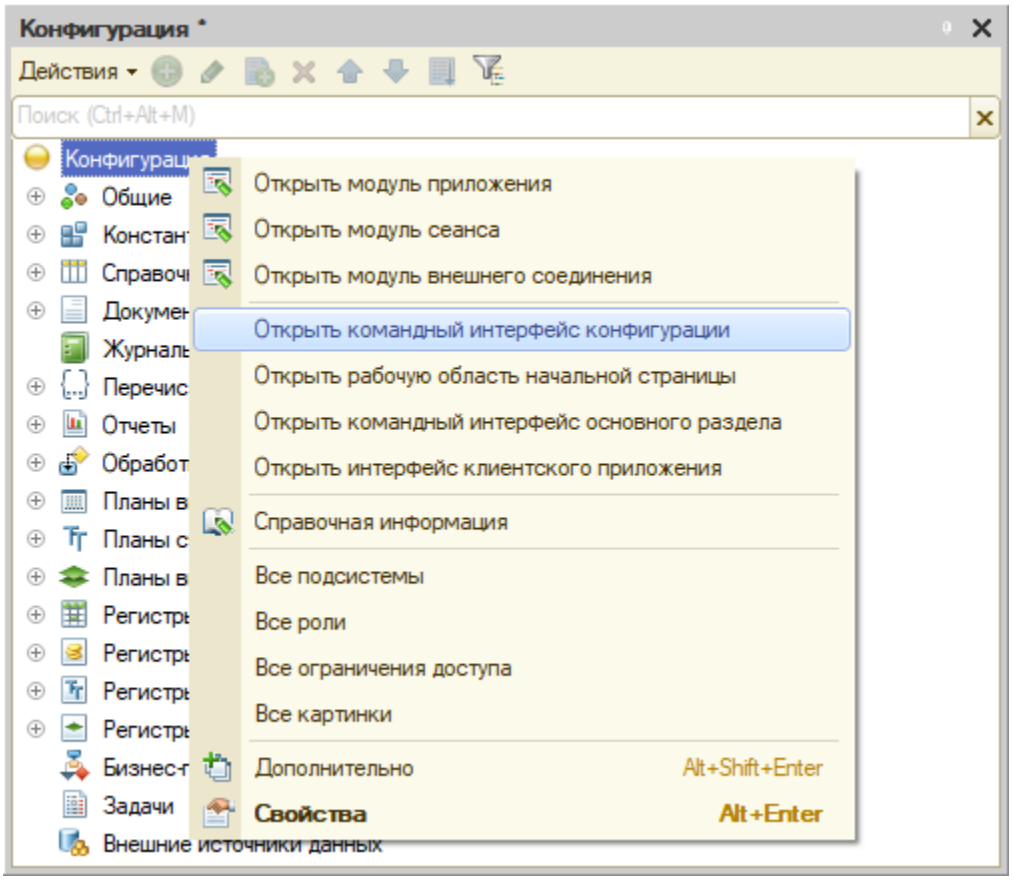


Рис. Вызов окна настройки командного интерфейса конфигурации.

Откроем редактор командного интерфейса конфигурации, содержащий список подсистем, и с помощью кнопок Вверх и Вниз поменяем порядок следования подсистем (рис.).

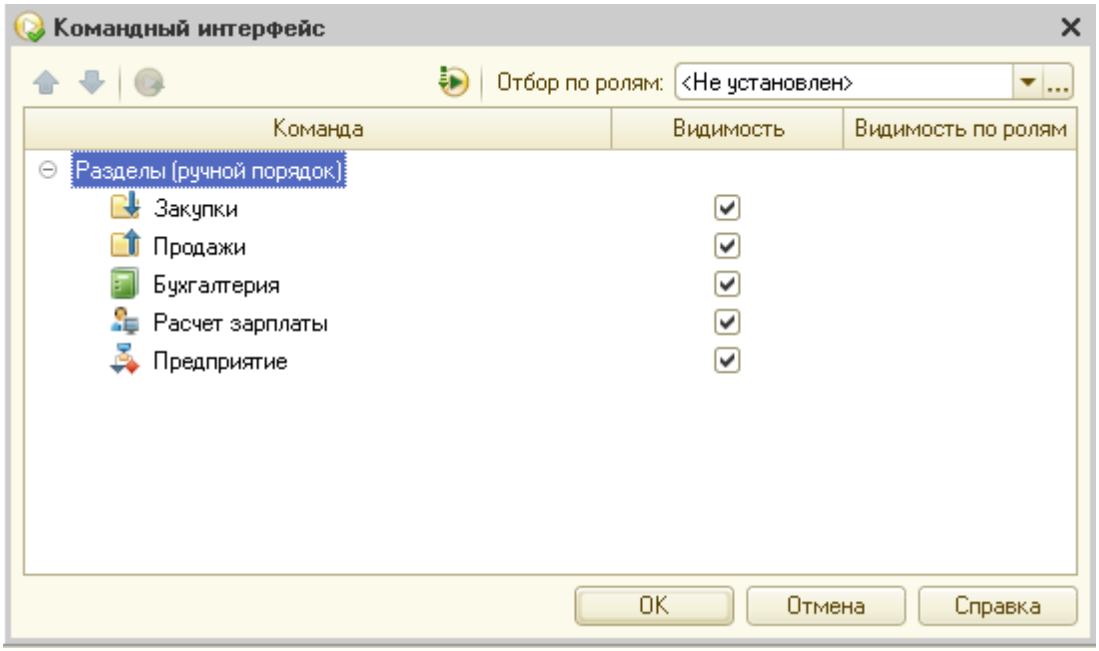


Рис. Настройка порядка следования подсистем с помощью редактора командного интерфейса

Откроем редактор интерфейса клиентского приложения. В левой части окна перечислены возможные места расположения панелей, в правой – доступные панели. Как видим, по умолчанию в верхней части окна расположены панель разделов и панель функций текущего раздела.

Перетащим панель открытых на пиктограмму Низ (рис.).

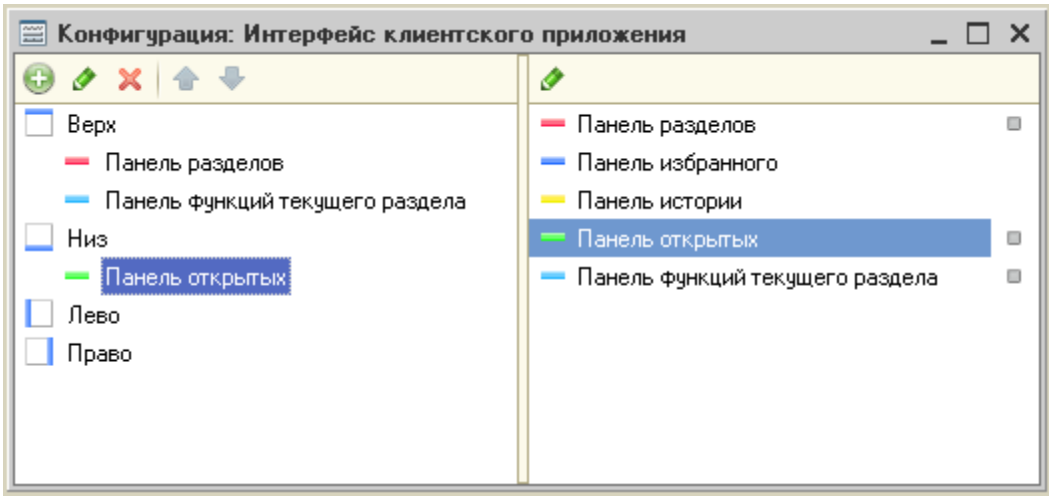


Рис. Окно редактора интерфейса клиентского приложения

Откроем редактор Все подсистемы одним из перечисленных в таблице способов (рис.).

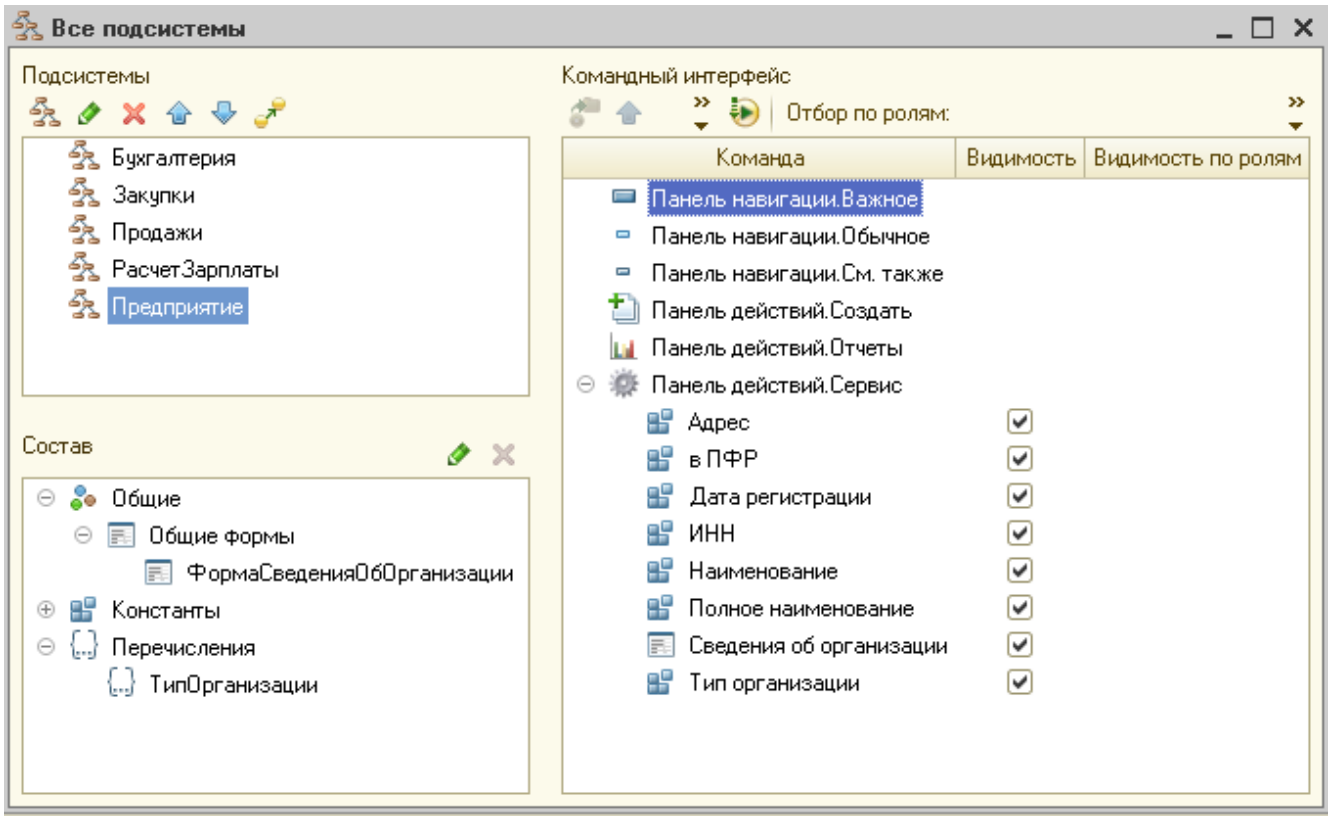


Рис. Окно редактора Все подсистемы

Окно редактора поделено на 3 области. Верхний левый список позволяет управлять составом и порядком расположения подсистем. Нижний левый список предназначен для редактирования состава каждой подсистемы. В правой области расположена таблица, позволяющая настраивать порядок и видимость команд по ролям.

Из таблицы Командный интерфейс видно, что команды в панели команд подсистемы по умолчанию располагаются в следующем порядке:

- *Панель навигации.Важное* содержит команды перехода к наиболее важным или наиболее часто используемым объектам конфигурации (справочникам, документам и т.д.); названия команд из данного раздела выделяются жирным шрифтом;
- *Панель навигации.Обычное* содержит команды обычного приоритета, которые жирно не выделяются;
- *Панель навигации.См.также* содержит наименее часто используемые команды;
- *Панель действий.Создать* объединяет в группу Создать команды создания новых экземпляров объектов конфигурации;
- *Панель действий.Отчеты* объединяет в группу Создать команды открытия форм с отчетами;
- *Панель действий.Сервис* объединяет в группу Создать команды открытия форм с обработками.

Для выделенной подсистемы Предприятие видим, что в ветви Панель действий.Сервис перечислены команды открытия форм констант данной подсистемы и они все видимы по умолчанию. Но поскольку мы создали новую форму для отображения всех констант, то отключим видимость для остальных форм (рис.).

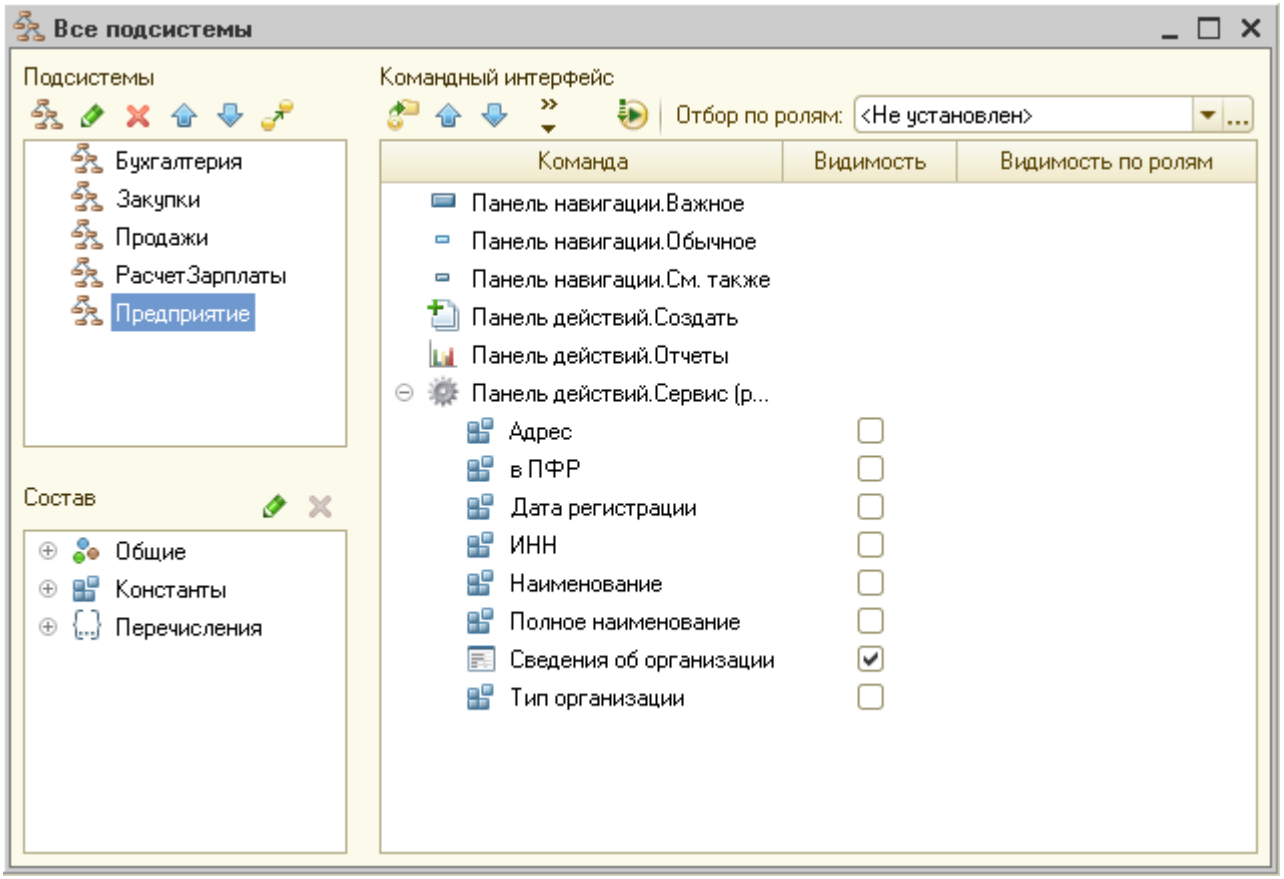


Рис. Настройка панели команд подсистемы Предприятие

Если теперь запустить приложение, то в меню Сервис подсистемы Предприятие отобразится только команда открытия формы Сведения об организации.

Настройка интерфейса приложения в режиме 1С: Предприятие

Настройка интерфейса приложения может производиться не только в режиме конфигуратора, но и непосредственно пользователем. Для этого используются команды, объединенные в группу Настройки, доступную при выборе команды Сервис и настройки на панели быстрого доступа в правой верхней части окна (рис.). Однако в отличие от настройки интерфейса в конфигураторе, изменения в настройках интерфейса в приложении будут выполнены только для текущего пользователя.

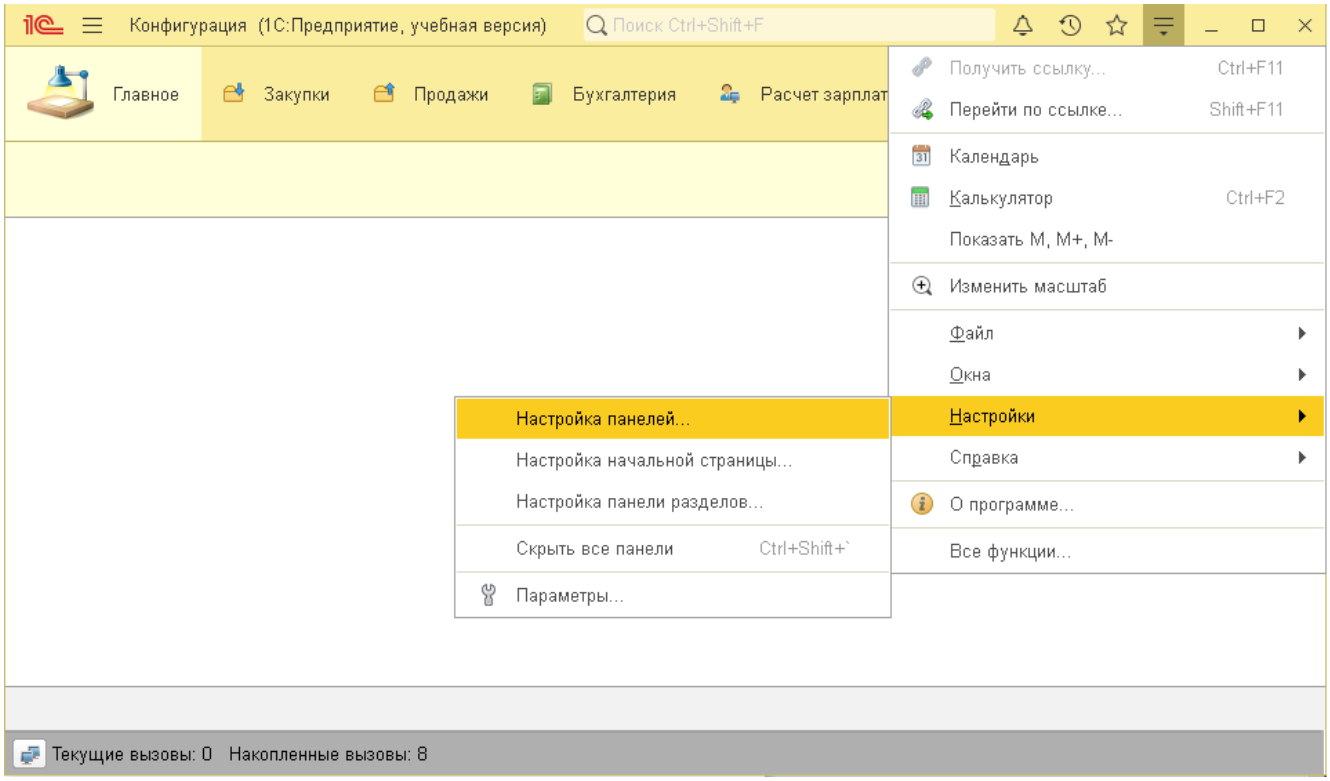


Рис. Команды настройки интерфейса приложения

При нажатии команды *Настройка панелей* открывается редактор панелей (рис.), который позволяет дополнить окно приложения панелями избранного, истории и открытых для быстрого доступа к наиболее важным, недавно открытыми и открытыми в настоящее время формам.

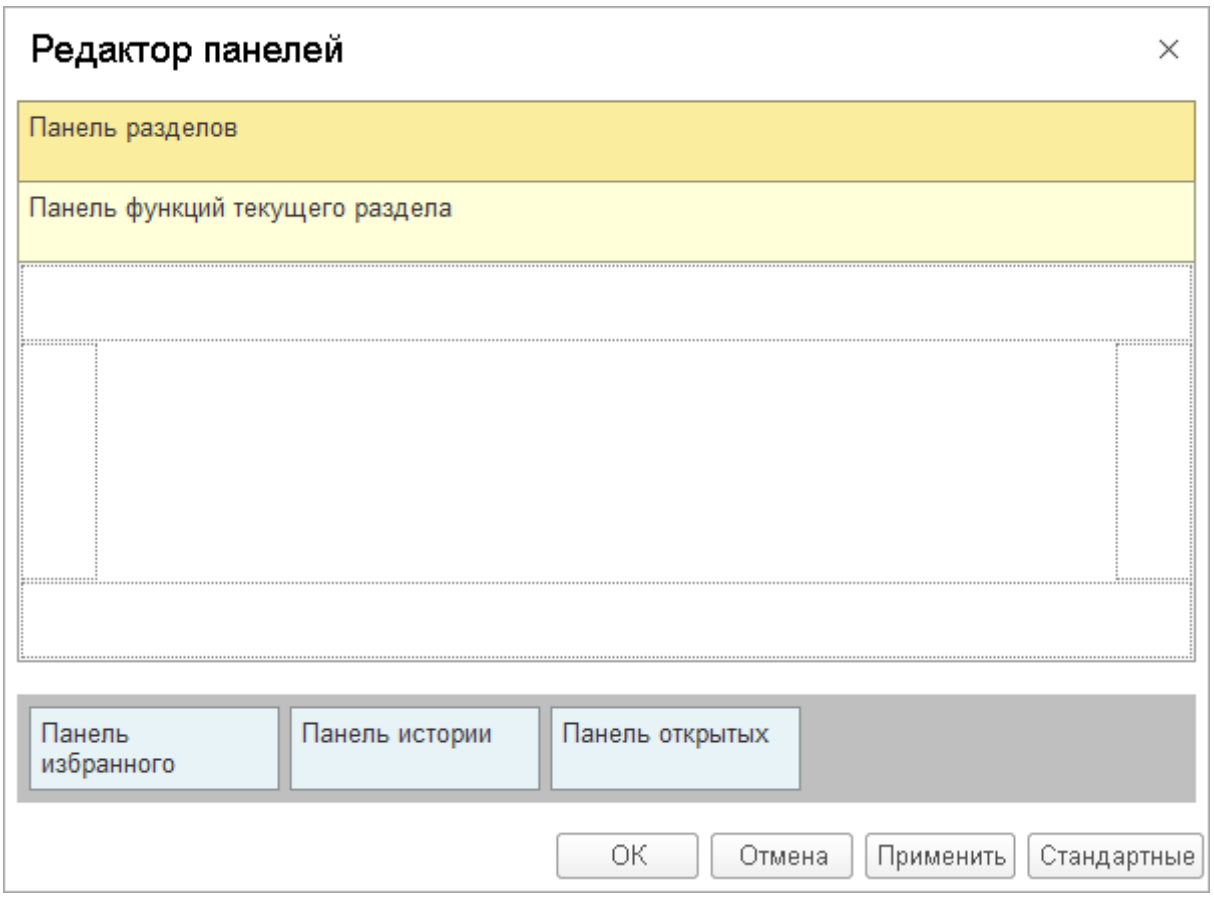


Рис. Редактор панелей

Команда Настройка панелей разделов позволяет управлять порядком расположения подсистем в панели разделов. При этом порядок настраивается только для текущего пользователя.

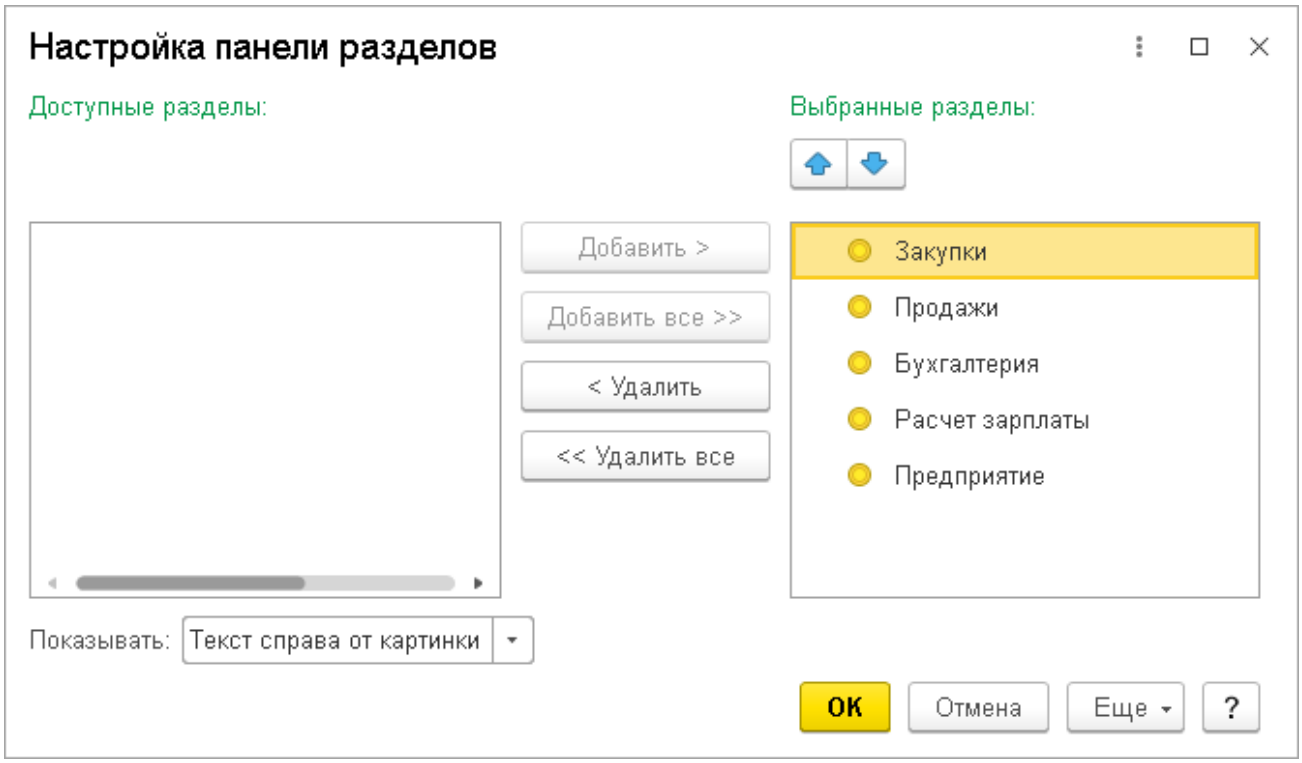


Рис. Диалоговое окно настройки панели разделов

Загрузка и выгрузка информационной базы

Если нужно сохранить информационную базу, то надо в режиме конфигуратора выбрать команду Администрирование / Выгрузить информационную базу и указать путь сохранения базы. Все данные конфигурации сохранятся в файле формата *.dt. Для того чтобы открыть базу на другом компьютере, необходимо создать новую пустую конфигурацию (или открыть уже существующую) и выполнить команду Администрирование / Загрузить информационную базу и выбрать ранее сохраненный файл.

Задание на лабораторную работу

1. Выполнить все примеры, рассмотренные в лабораторной работе.
2. Найти в открытых источниках описание типовых конфигураций 1С (1С: ERP, 1С: Бухгалтерия и т.д.). Сохранить скриншот формы со сведениями об организации из типовой конфигурации.
3. По примеру формы из типовой конфигурации добавить еще минимум 4 константы для хранения кодов и регистрационных номеров организации.
4. Добавить константы Телефон и E-mail.
5. Задать маски ввода для констант. Предусмотреть возможность ввода значений разного формата для одной константы (например, стационарного и мобильного номеров телефона). Обосновать, для каких констант создавать маску ввода не требуется.
6. Добавить константы на форму Сведения об организации.
7. Скрыть команды открытия стандартных форм констант из меню Сервис.
8. Выгрузить информационную базу и сохранить ее для следующей лабораторной работы.

Лабораторная работа №3-4
Справочники

Справочники

Справочники в 1С: Предприятие предназначены для хранения однородной информации списочного характера, например, списка товаров, сотрудников и т.д. В дальнейшем данные из справочника могут подгружаться как список возможных значений для реквизитов других объектов конфигурации, например, документов. В отличие от перечислений, редактировать элементы которых может только разработчик, элементы справочника может создавать и изменять пользователь приложения.

Справочник представляет собой таблицу, состоящую из столбцов, называемых в терминах 1С полями или реквизитами, и строк – элементов справочника.

Каждый справочник имеет набор стандартных реквизитов, создаваемых ему системой автоматически. К наиболее важным стандартным реквизитам справочника относятся код и наименование. *Код* – уникальный идентификатор элемента справочника. Он может быть как числовым, так и текстовым. Система поддерживает режим автоматической генерации кода, при котором она самостоятельно может генерировать код для нового элемента справочника. Кроме этого система позволяет осуществлять контроль уникальности кодов справочника, не разрешая создавать элементы с одинаковыми кодами.

Если кода и наименования недостаточно, то разработчик может создать дополнительные реквизиты и табличные части для хранения таблицы значений.

Создание справочника

Создадим справочник Контрагенты, содержащий наименования организаций, с которыми взаимодействует наша фирма. Для этого в дереве объектов конфигурации выделим ветвь Справочники и нажмем кнопку Добавить. В открывшемся окне редактирования объекта конфигурации зададим имя справочника – Контрагенты. На основании имени платформа автоматически создаст синоним – Контрагенты. Зададим представление объекта Контрагент (рис.).

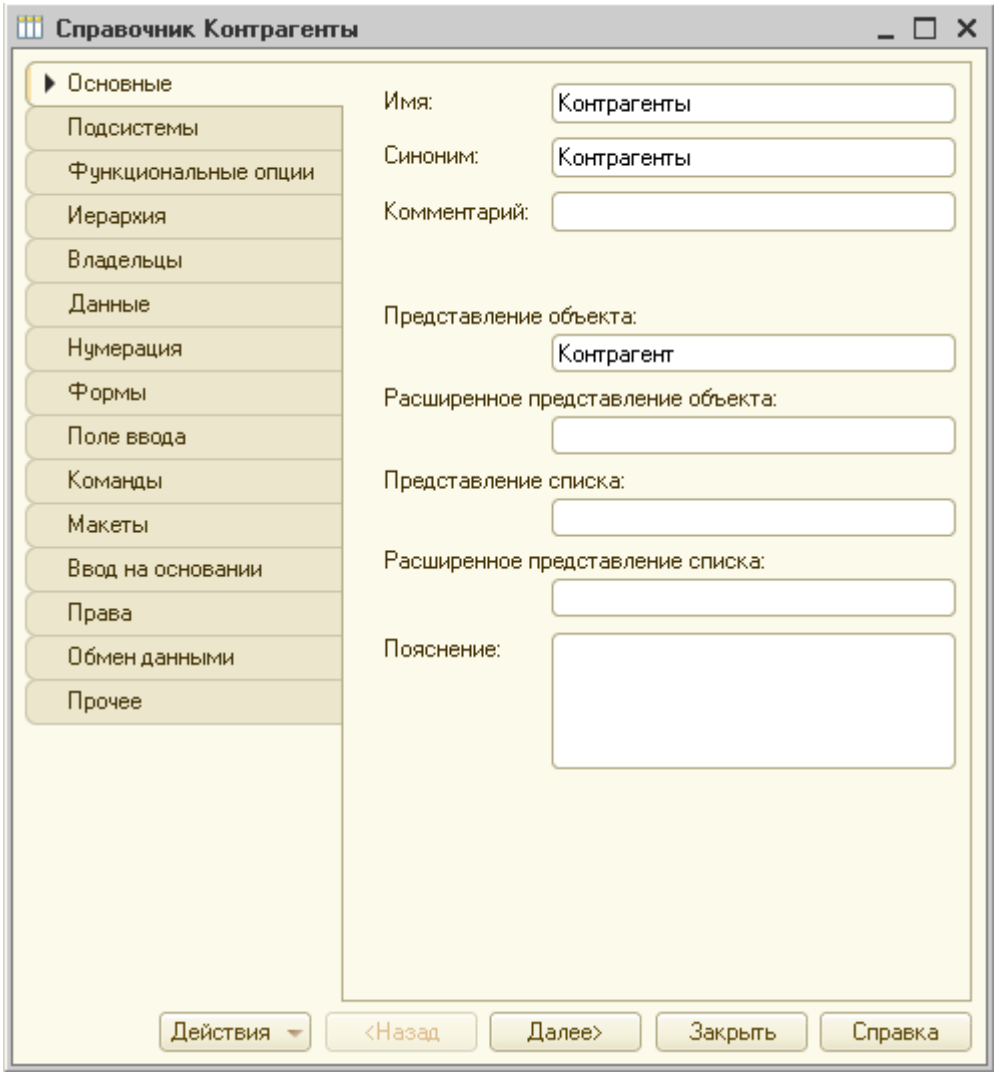


Рис. Создание справочника Контрагенты

На вкладке Подсистемы отметим разделы Предприятие и Бухгалтерия (рис.).

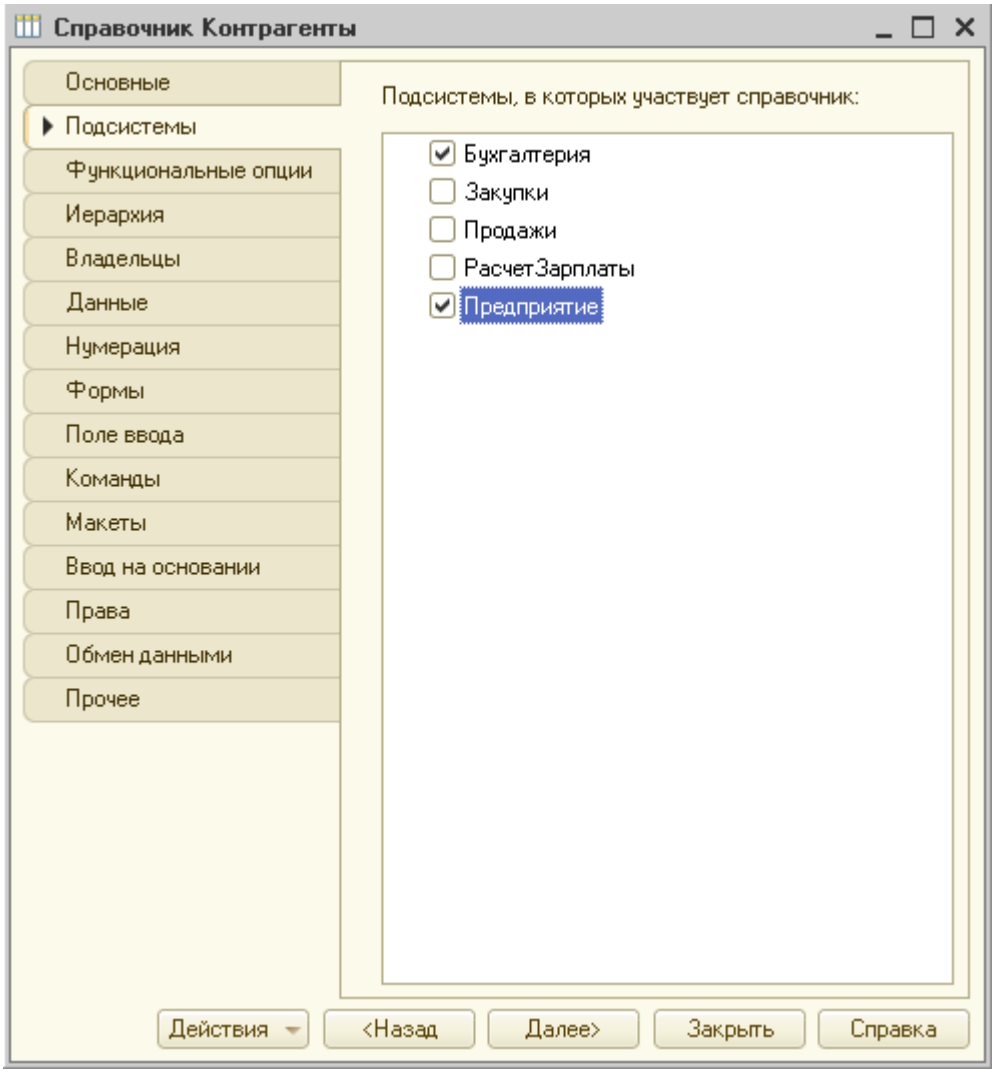


Рис. Выбор подсистем, в которые будет входить справочник

Перейдем на закладку Данные. Здесь для нас представляют интерес длина кода и длина наименования.

Длина кода – важное свойство справочника. Как правило, код справочника используется для идентификации элементов справочника и содержит уникальные для каждого элемента справочника значения. Платформа может сама контролировать уникальность кодов и поддерживать автоматическую нумерацию элементов справочника. Поэтому от длины кода будет зависеть количество элементов, содержащихся в справочнике. По умолчанию установлена длина кода 9 символов, следовательно, можно использовать коды от 1 до 999999999.

Длина наименования – длина строки для названия конкретного экземпляра объекта конфигурации. По умолчанию установлена длина 25 символов.

Увеличим длину наименования до 50, чтобы строки точно хватило для названия любой должности (рис.).

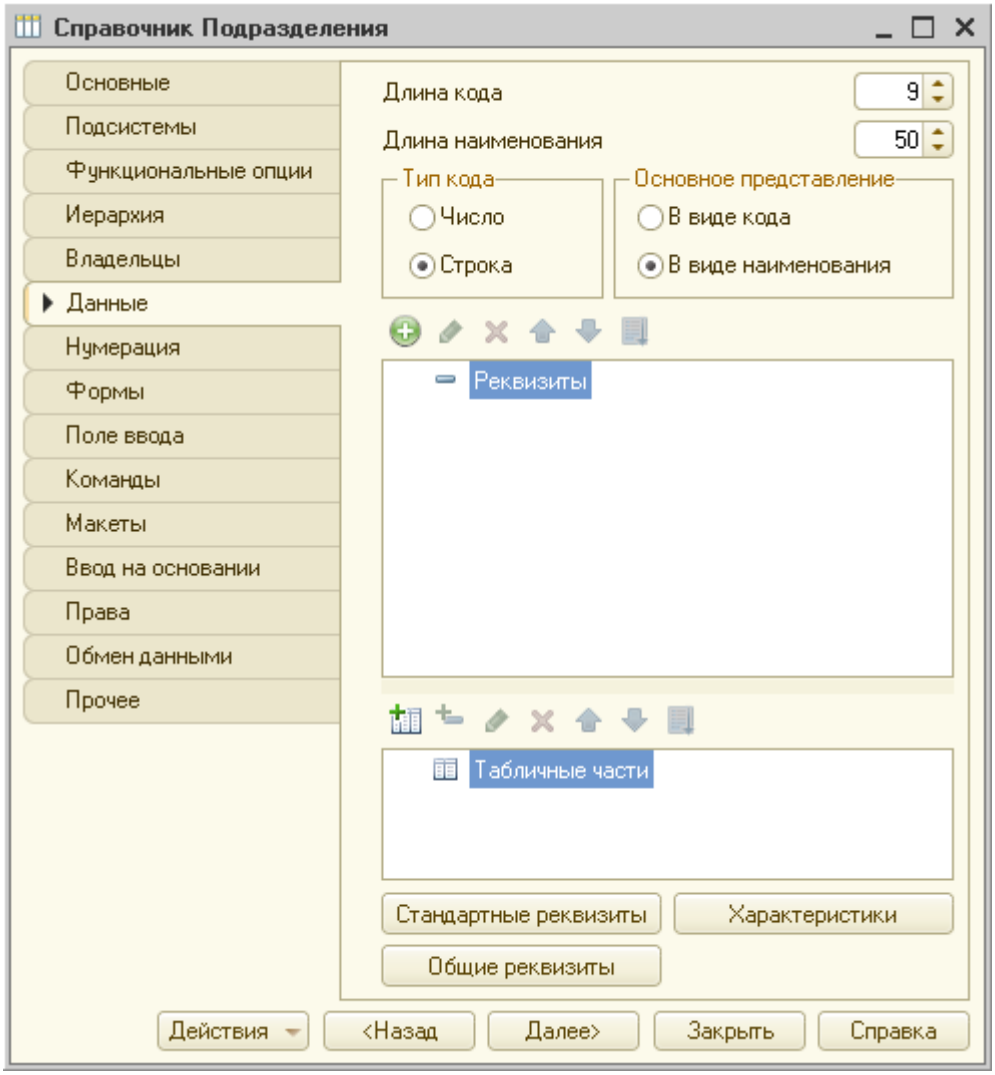


Рис. Вкладка Данные редактора справочника

Поскольку справочник должен хранить только список контрагентов, то пока нам будет достаточно стандартных реквизитов Код и Наименование.

Представления объекта конфигурации

Синоним объекта конфигурации, задаваемый в конфигураторе, используется в элементах интерфейса приложения (заголовках окон, списков и т.д.). Но иногда требуется задать имена объектов, отличные от синонима. В этом случае используются представления.

Представление объекта определяет название объекта в единственном числе и используется в названии стандартной команды, например, команды создания объекта – Клиент: создать.

Расширенное представление объекта определяет заголовок формы объекта, например формы для создания нового элемента справочника. Если это свойство не задано, то вместо него используется свойство Представление объекта.

Представление списка определяет название списка объектов и используется в названии стандартной команды.

Расширенное представление списка определяет заголовок формы списка, например формы списка справочника. Если это свойство не задано, то вместо него используется свойство Представление списка.

Добавление, редактирование и удаление элементов справочников в режиме 1С:Предприятие

Запустим приложение в режиме отладки и откроем подсистему Предприятие. Под наименованием раздела располагается список команд текущего раздела. Пока что в панели команд видим только команду открытия списка для только что добавленного справочника Контрагенты. Заметим, что имя команды Контрагенты определяется свойством Представления списка (вкладка Основное окна редактирования справочника). Поскольку мы не заполняли данное значение, то оно совпадает с синонимом Подразделения.

Для добавления элементов в справочник можно использовать один из следующих способов:

- нажать кнопку Создать в форме списка Подразделения;
- нажать на клавиатуре кнопку Insert (Ins).

После выполнения любого из перечисленных действий откроется стандартная форма для создания элемента справочника (рис.).

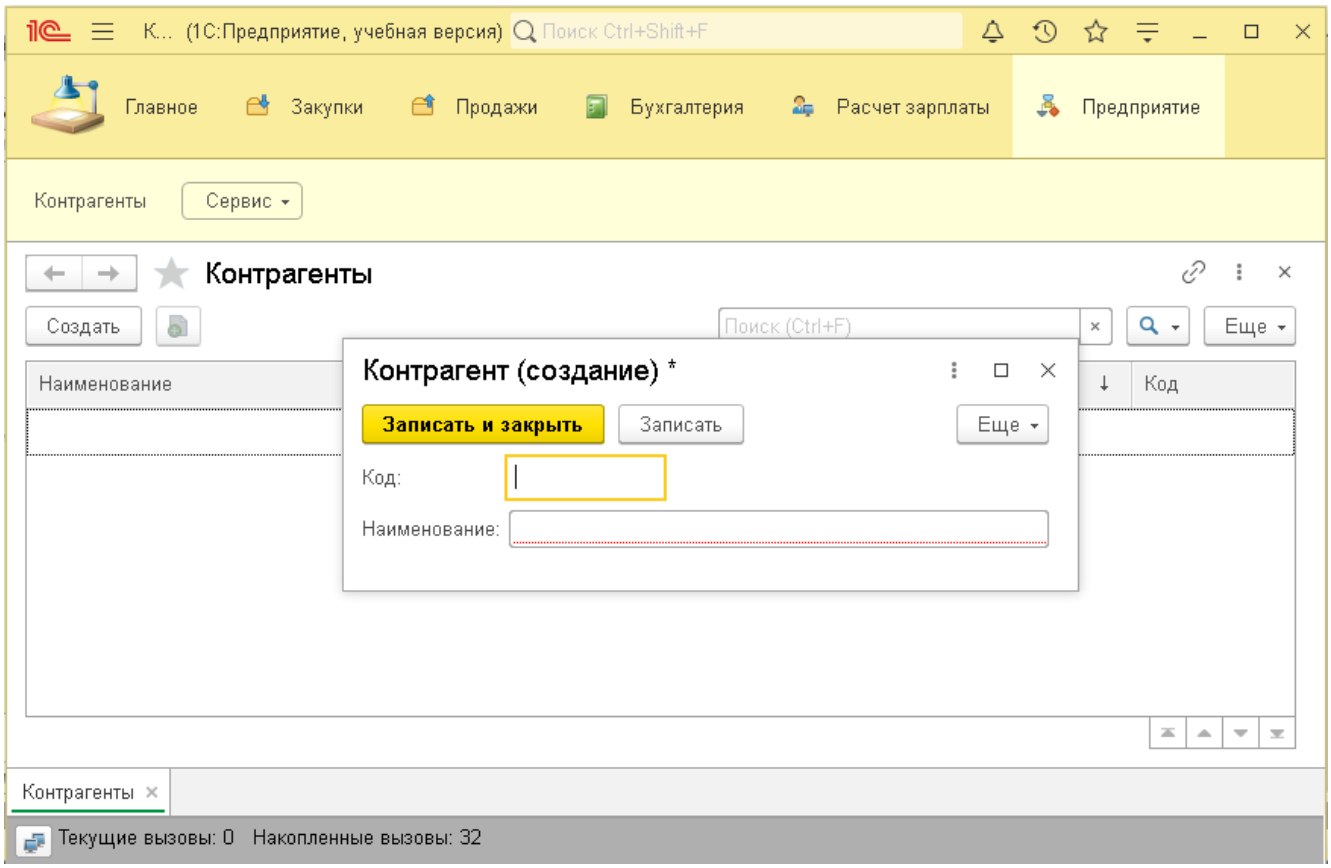


Рис. Форма создания нового элемента справочника Контрагенты

Заголовок формы Контрагент (создание) определяется расширенным представлением объекта. Мы это поле оставили пустым, поэтому вместо него используется представление объекта, где мы и указали название Контрагент.

Обратим внимание, что поле Наименование формы выделено красной пунктирной линией. Это означает, что данное поле должно быть обязательно заполнено. Если его оставить пустым, то будет выведено сообщение об ошибке.

Поле Код заполнять не будем, поскольку оно генерируется автоматически. Зададим Наименование нового контрагента ООО Маркер и нажмем кнопку Записать и закрыть. При этом в нижнем правом углу экрана появится сообщение о том, какой элемент был создан, и клиент отобразится в списке.

Если нужно отредактировать элемент справочника, то надо дважды щелкнуть на нем мышью. После этого откроется окно редактирования элемента.

Процедура удаления элемента справочника выполняется в два этапа:

- пометка пользователем элемента на удаление;
- удаление элементов, помеченных на удаление, администратором системы.

Такие действия необходимы для того, чтобы не была нарушена ссылочная целостность, поскольку удаляемый элемент может использоваться другими объектами базы.

Для того чтобы пометить элемент справочника на удаление, необходимо выделить его в списке и выбрать команду Пометить на удаление в контекстном меню или списке команд Еще. Кроме этого можно на клавиатуре нажать кнопку Delete (Del). На рис. на удаление помечен элемент ООО Маркер (рис.).

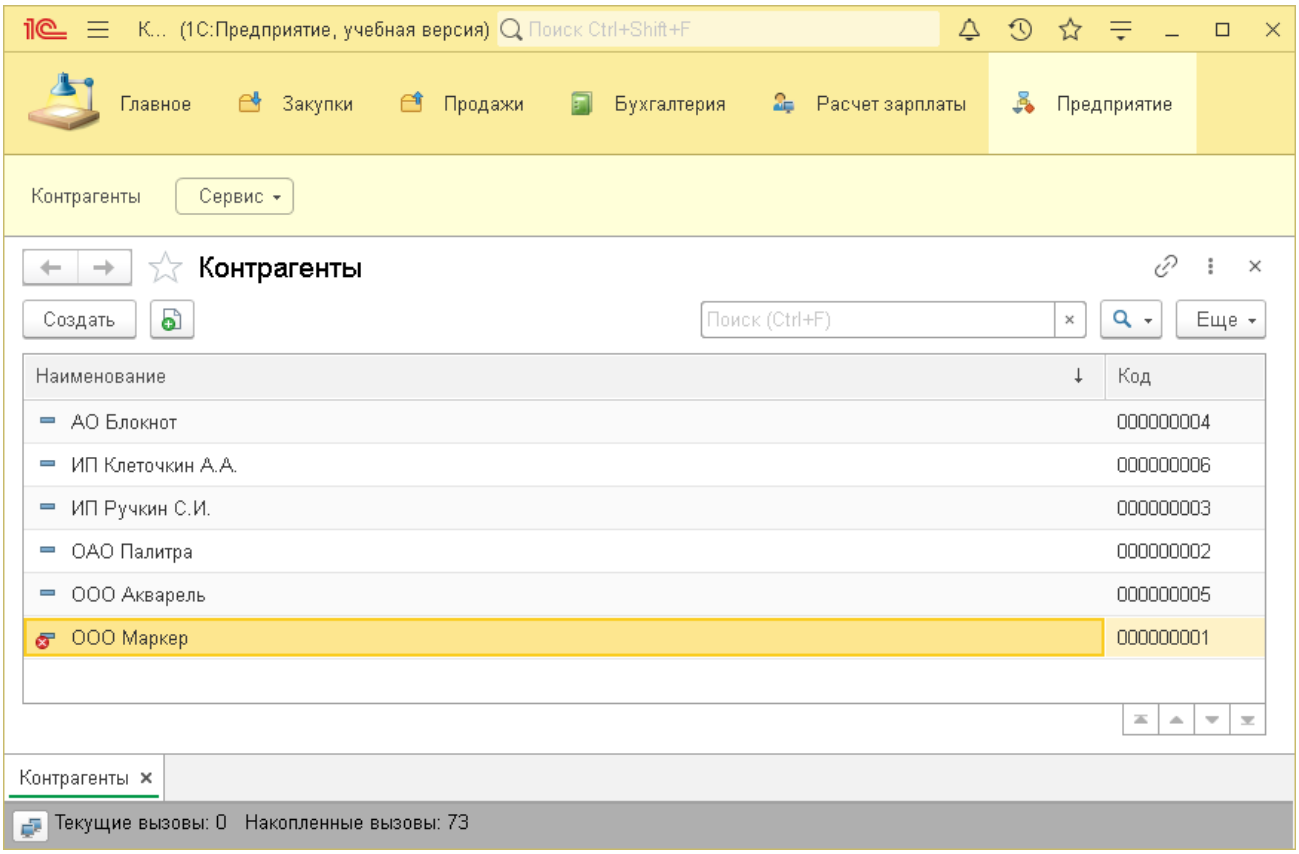


Рис. Установка пометки на удаление у элемента ООО Маркер

Чтобы теперь удалить помеченный элемент, надо в меню Еще выбрать команду Удалить.

Определение иерархии элементов справочника

Справочники в 1С: Предприятие могут поддерживать иерархическое расположение элементов. При этом система предусматривает два вида иерархии:

- иерархия групп и элементов - создаются группы, в которых располагаются элементы, относящиеся к этим группам; группы являются просто визуальными элементами и обладают ограниченным набором стандартных реквизитов; например, номенклатуру продаваемых фирмой товаров может быть разбита на группы Бумага, Тетради, Письменные принадлежности и т.д.

- иерархия элементов – все элементы справочника обладают одним и тем же набором реквизитов и одни элементы могут включать другие; например, все подразделения предприятия обладают своими функциями, при этом одни подразделения могут входить в состав других.

Создадим иерархический справочник Подразделения. В окне редактирования справочника зададим имя справочника – Подразделения, представление объекта - Подразделение. На вкладке Подсистемы отметим раздел Предприятие.

На вкладке Иерархия установим флажок Иерархический справочник и выберем вид иерархии Иерархия элементов (рис.).

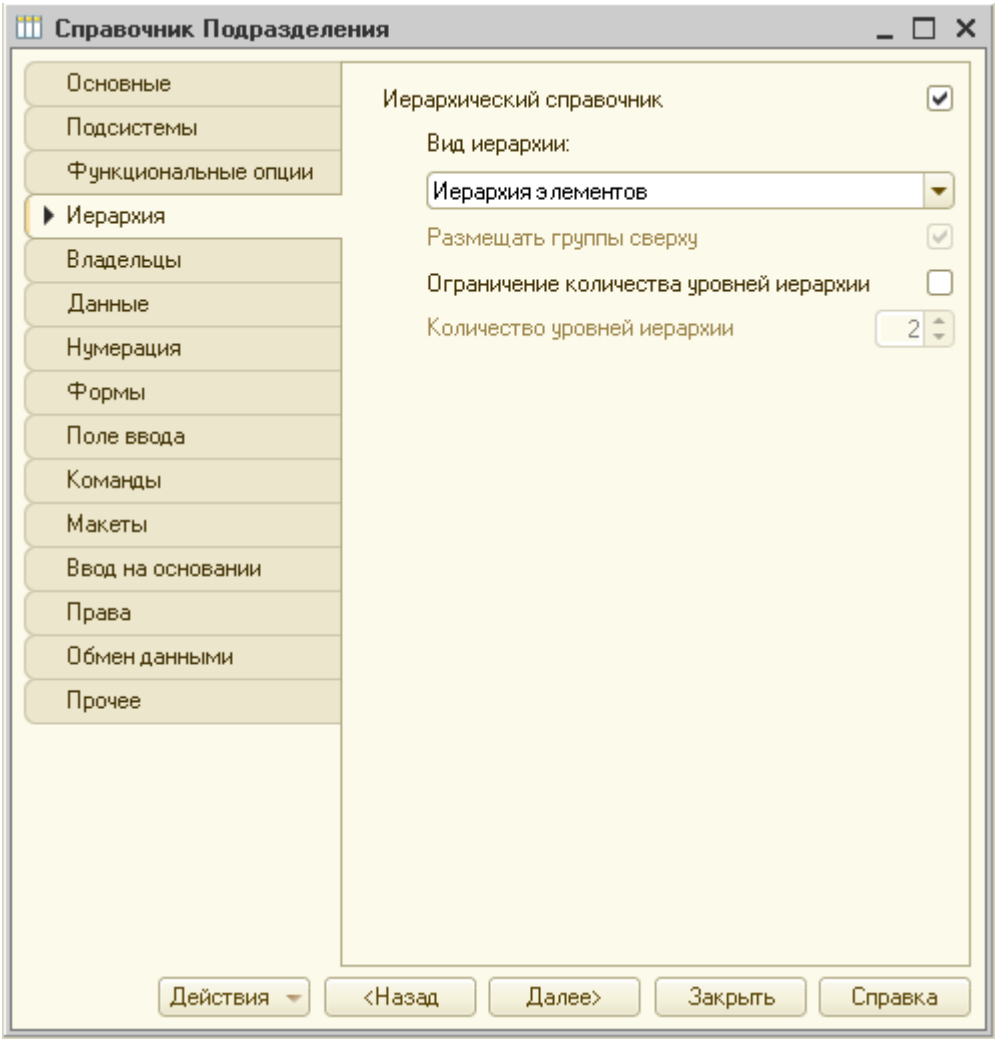


Рис. Определение вида иерархии справочника Подразделения

Запустим приложение в режиме отладки, откроем в подсистеме Предприятие справочник Подразделения и создадим подразделения верхнего уровня иерархии. В диалоговом окне создания нового элемента справочника появилось поле Родитель. Если его оставить пустым, то будет создан элемент первого уровня иерархии (рис.).

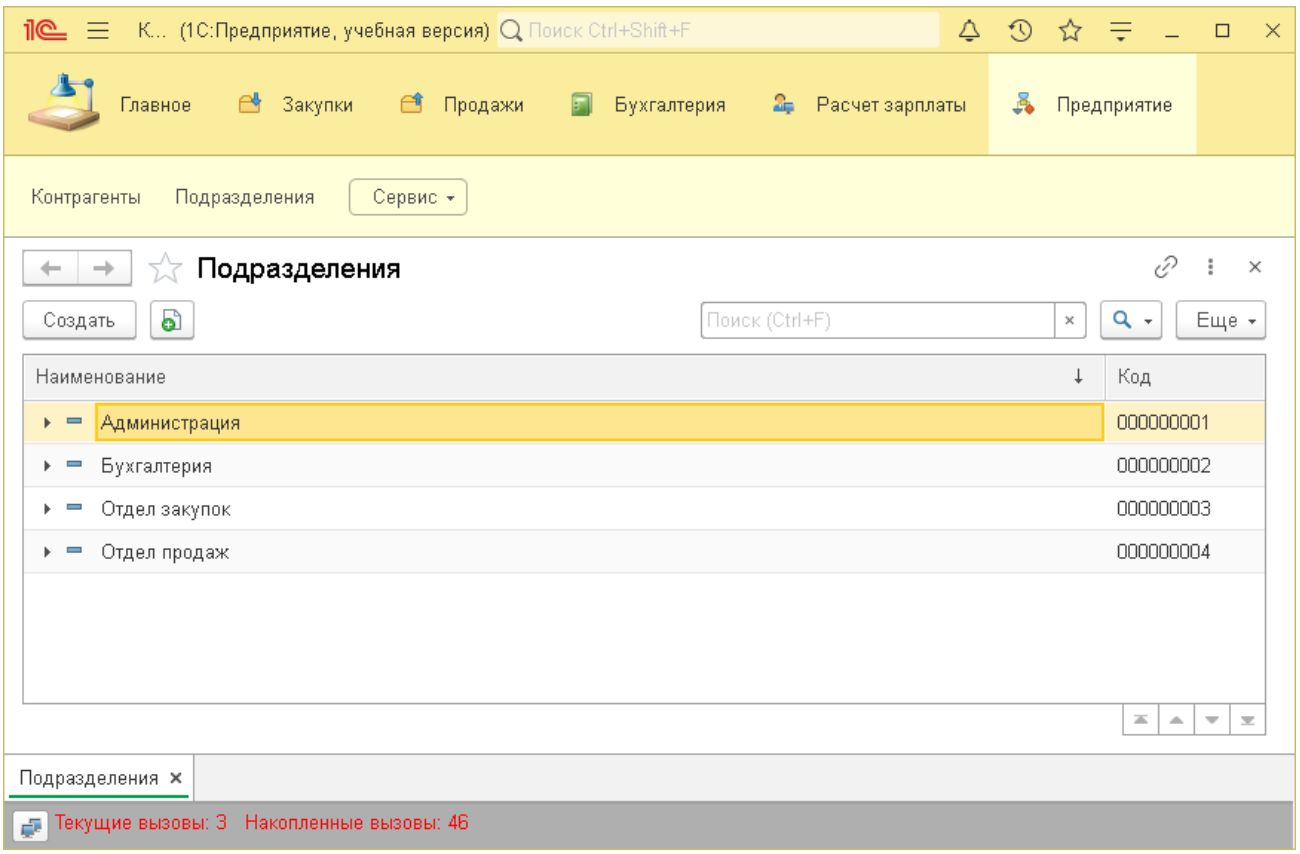


Рис. Список подразделений вернего уровня иерархии

Для подразделения Бухгалтерия создадим дочерние элементы Материальный отдел и Расчетный отдел (рис.).

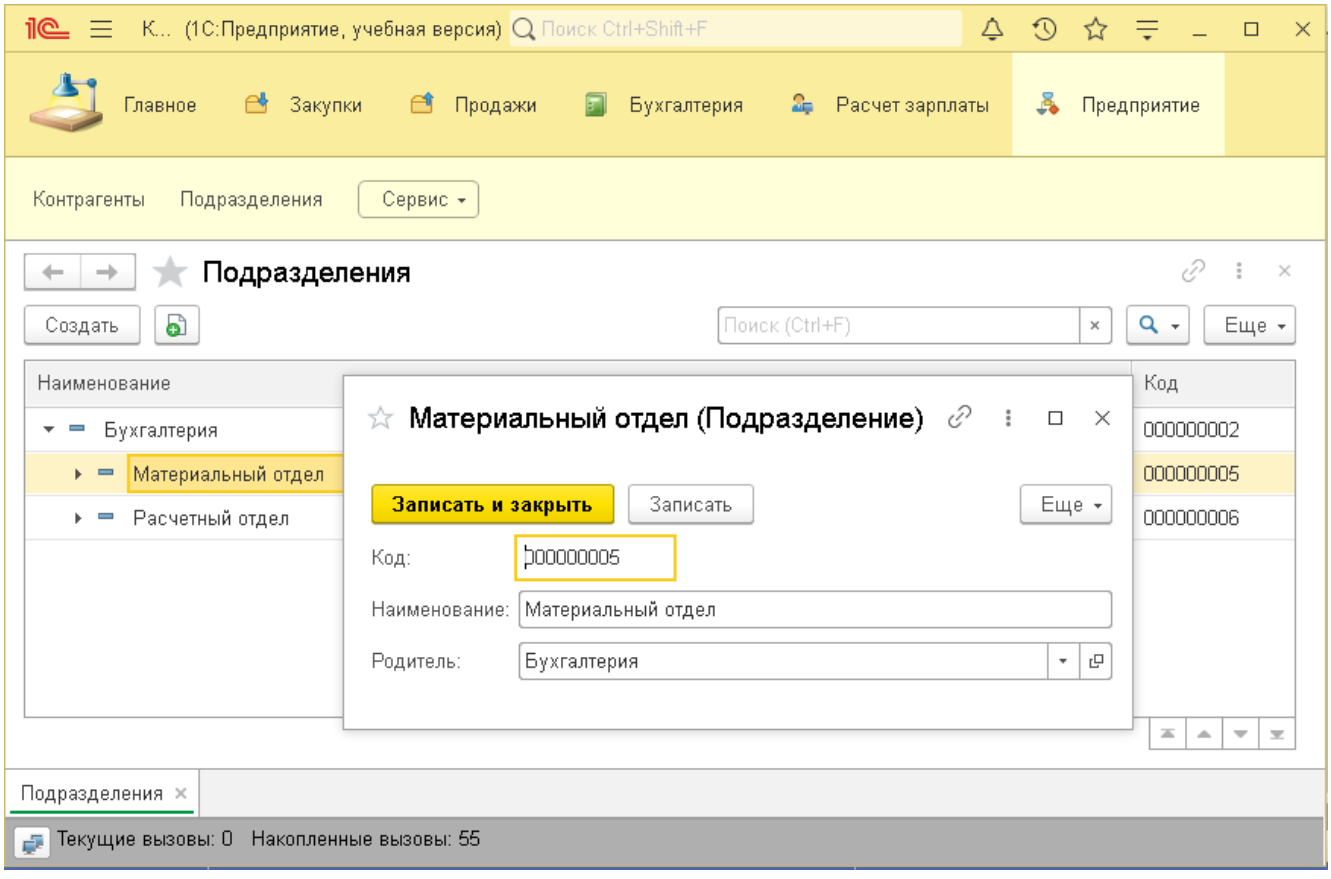


Рис. Создание дочерних подразделений

Настройка режима просмотра для иерархического справочника

Иерархический справочник имеет следующие режимы просмотра:

- иерархический список – на форме отображаются элементы только одного выбранного уровня иерархии;
- список – родительские и дочерние элементы отображаются одним списком;
- дерево – все уровни иерархии отображаются в виде дерева.

Чтобы изменить способ отображения элементов справочника, необходимо в меню Еще выбрать команду Режим просмотра (рис.).

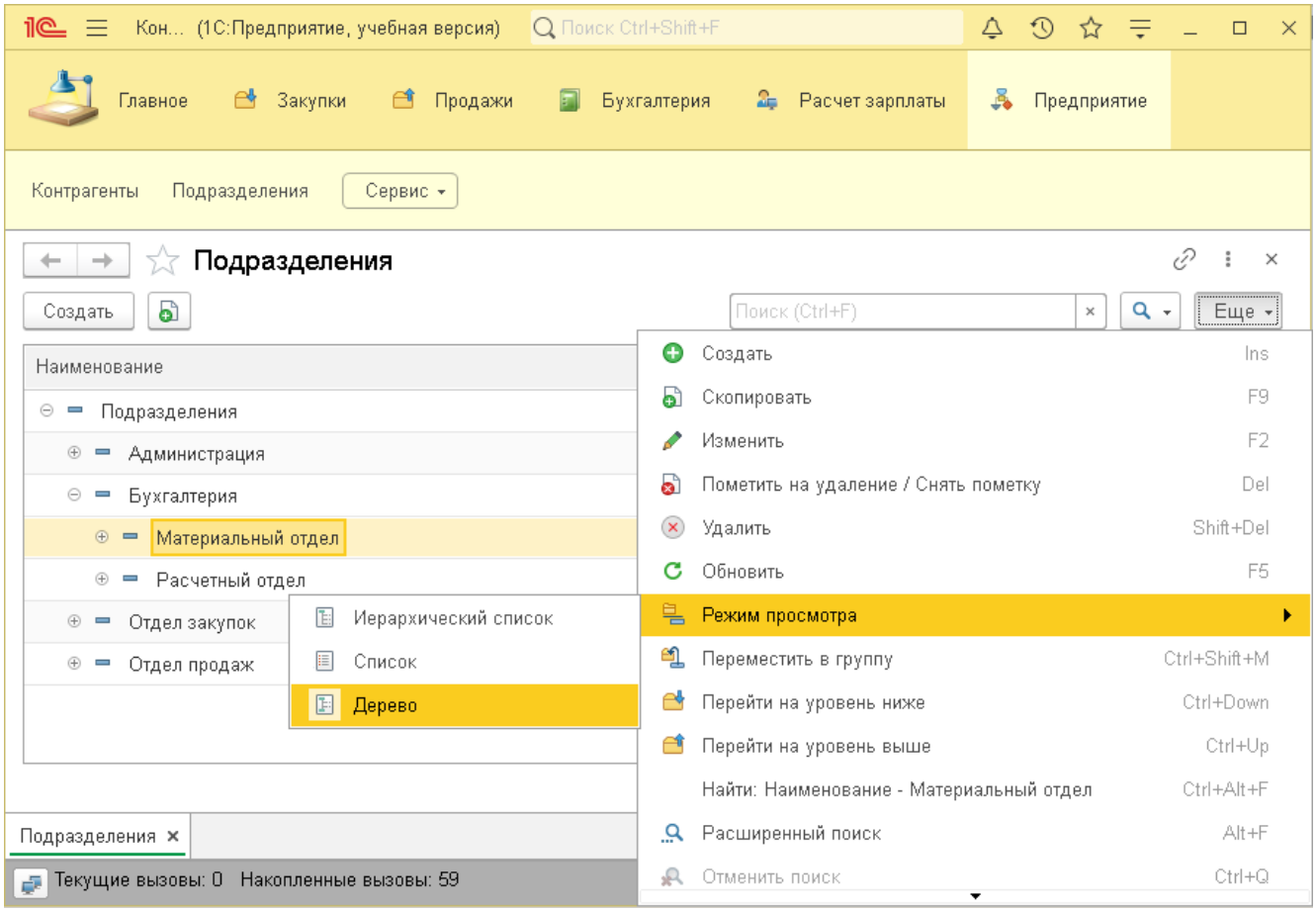


Рис. Выбор режима просмотра иерархического дерева

Создадим справочник Номенклатура, в котором будет храниться список реализуемых фирмой канцтоваров, но теперь уже зададим вид иерархии Иерархия групп и элементов (рис.).

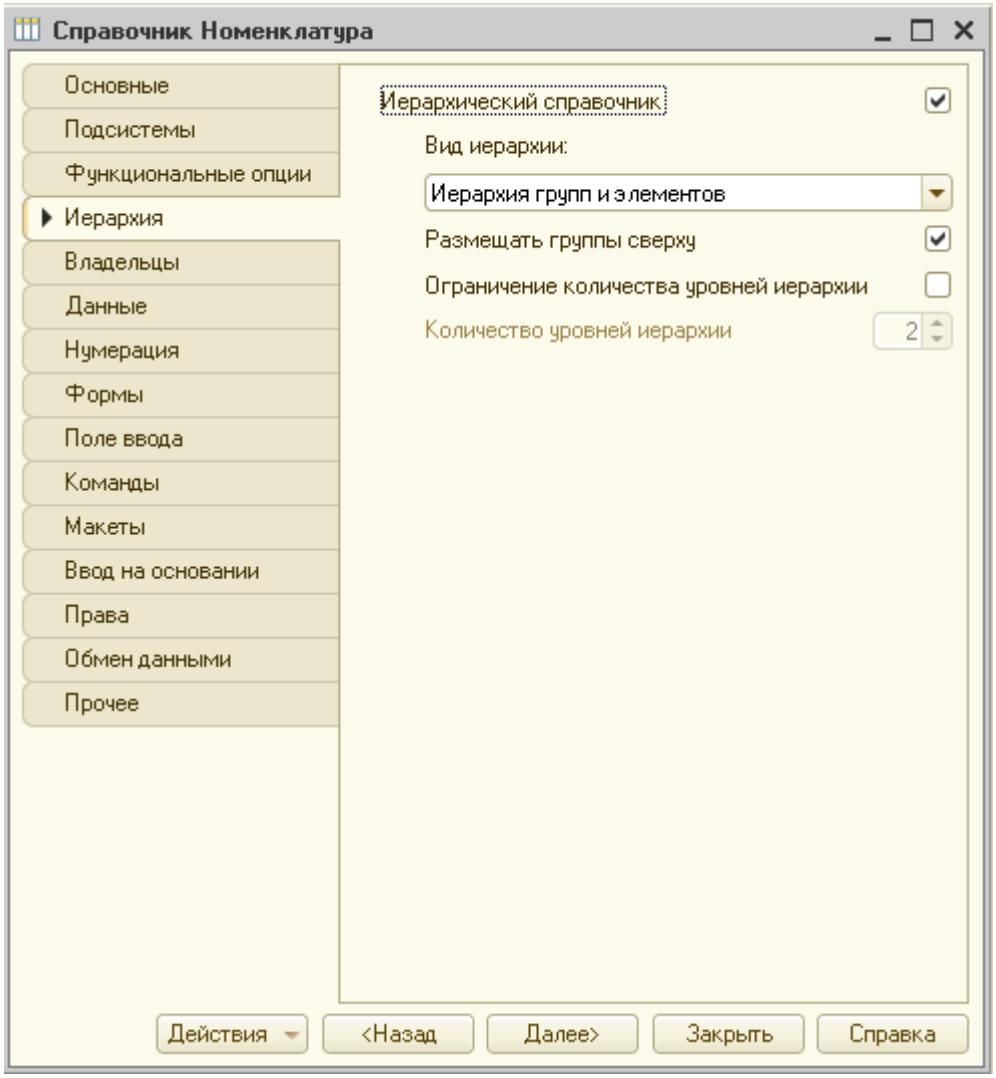


Рис. Определение вида иерархии для справочника Номенклатура

Заполним справочник в режиме отладки приложения (рис.).

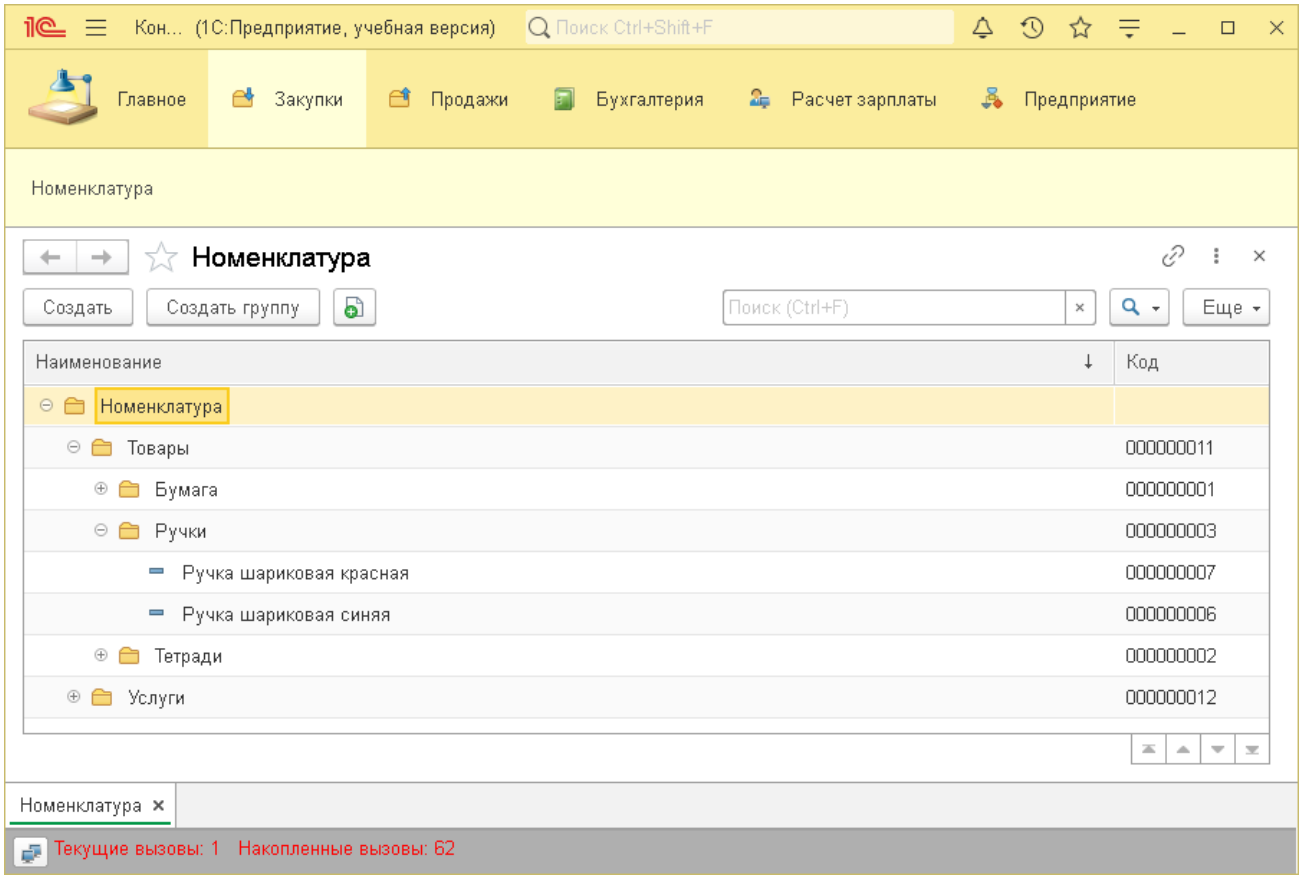


Рис. Заполнение справочника Номенклатура

Создание подчиненного справочника

В системе один справочник может быть подчинен другому. Это означает, что каждый элемент подчиненного справочника обязательно имеет владельца – элемент или группу из другого справочника. При этом один элемент «родительского» справочника может быть владельцем нескольких элементов из подчиненного справочника.

Такая возможность обычно используется для контроля ввода данных пользователем, когда при выборе значения в одном списке необходимо ограничить множество элементов в другом списке. Например, при выборе города в списке улиц надо оставить только улицы этого города.

Создадим справочник Должности. В учебных целях предположим, что в каждом подразделении свой список должностей.

В диалоге создания справочника на вкладке Основное зададим имя справочника и представление объекта Должность (рис.). На вкладке Подсистемы отметим подсистему Предприятие.

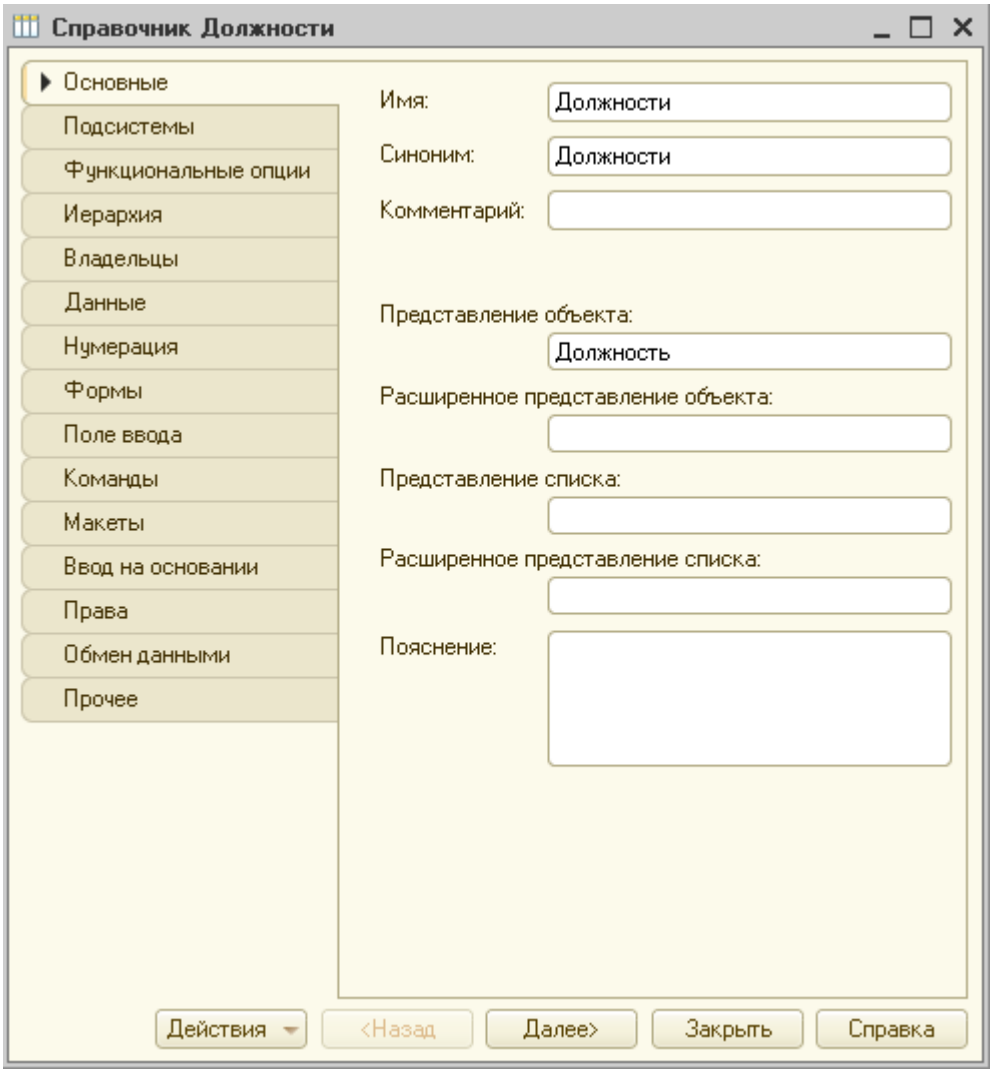


Рис. Создание справочника Должности

На вкладке Владельцы нажмем на кнопку Редактировать элемент списка ( ) и выберем в списке справочник Подразделения (рис.).

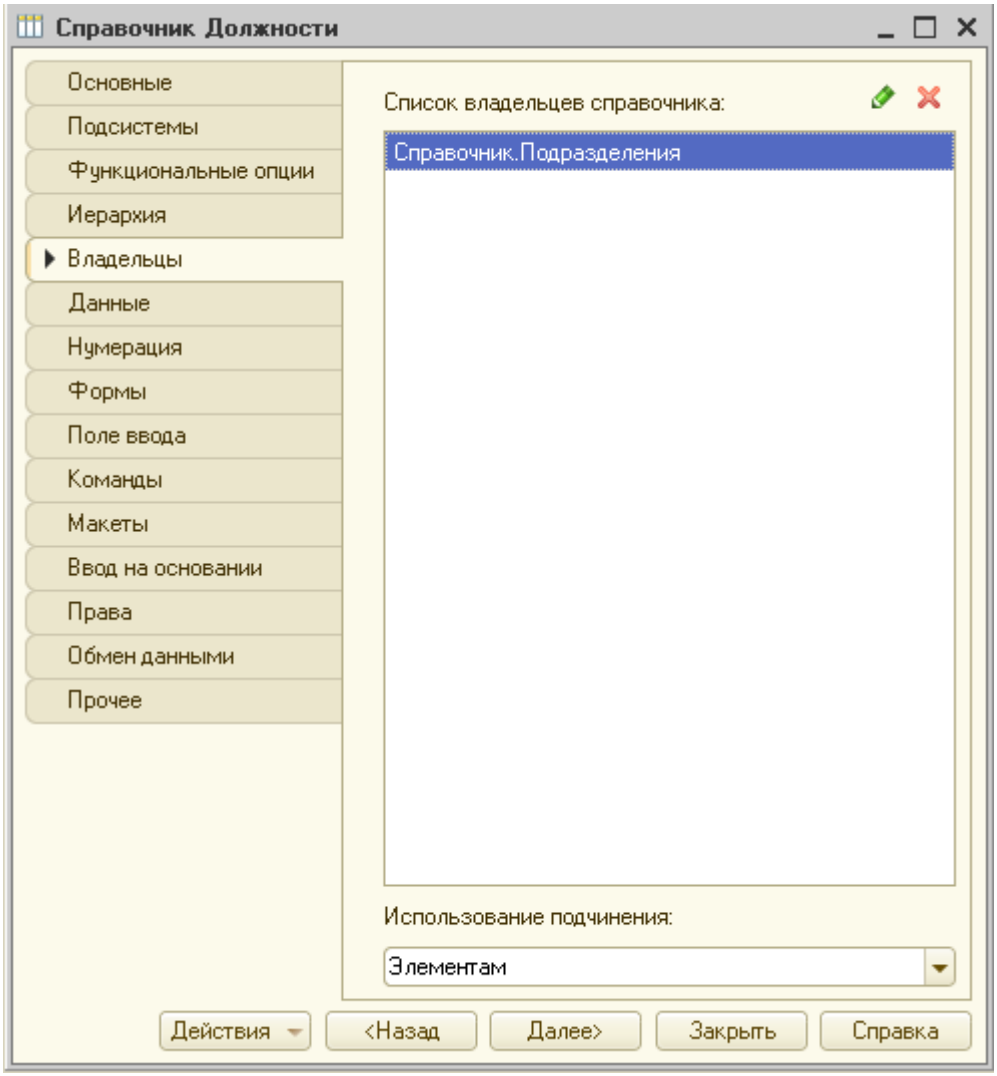


Рис. Определение справочника-владельца

Запустим приложение в режиме 1С: Предприятие, откроем раздел Предприятие, справочник Должности и создадим новый элемент справочника (рис.).

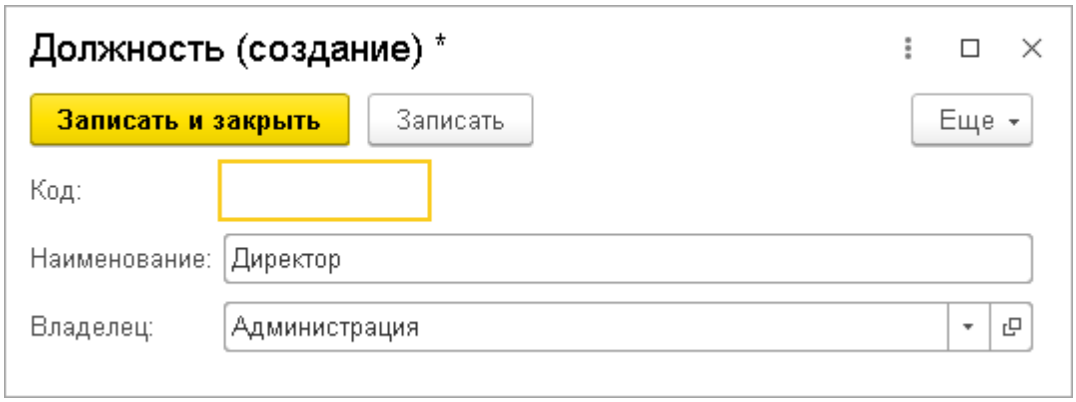


Рис. Создание элемента справочника Улицы

Как видно из рис. в диалоговом окне создания элемента справочника появилось обязательное для заполнения поле Владелец. Добавим в справочник еще несколько элементов (рис.).

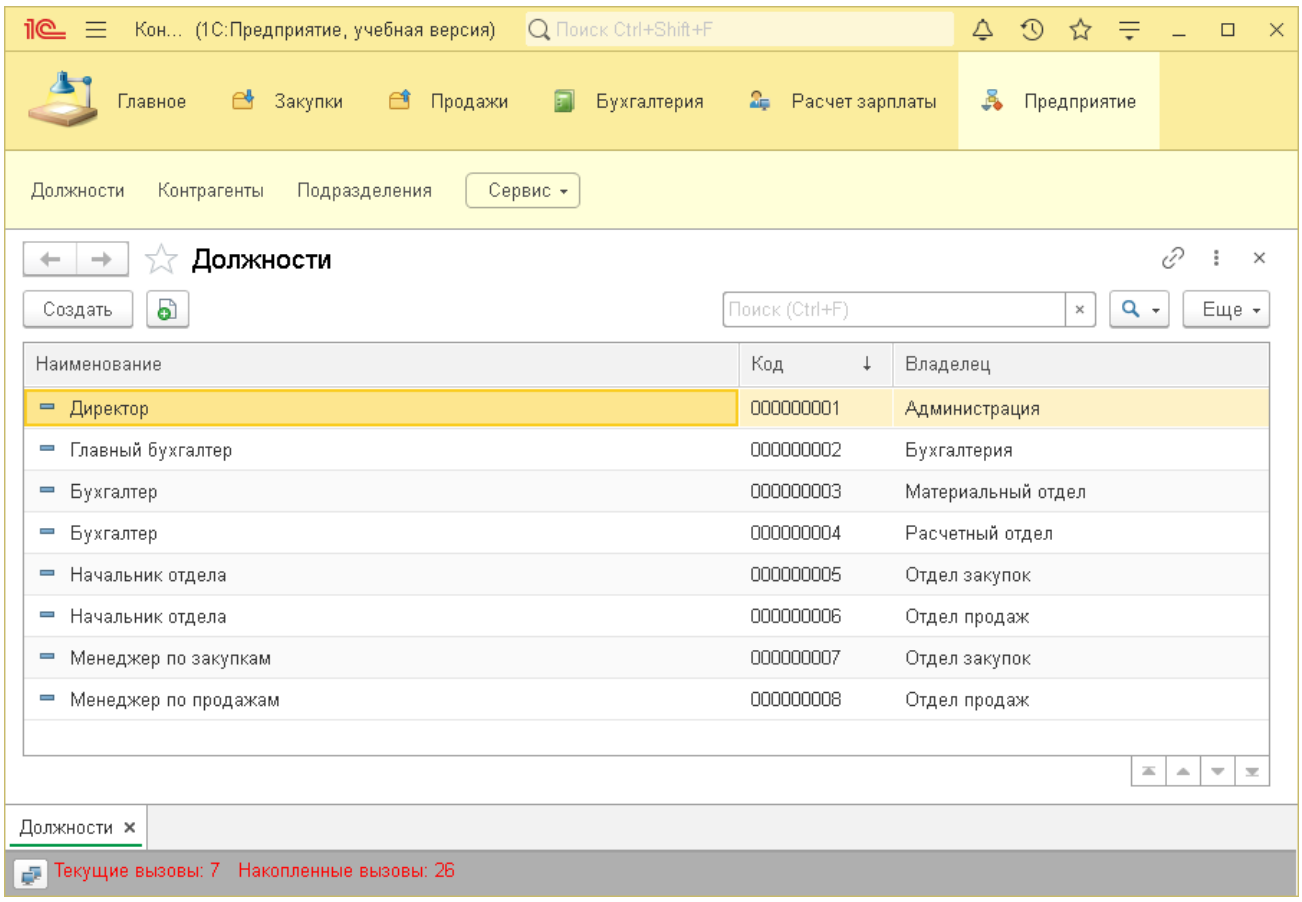


Рис. Справочник Должности

Теперь откроем форму списка элементов справочника Подразделения. После определения подчиненного справочника на форме создания / редактирования элементов родительского справочника появилась команда-ссылка на список подсиненных элементов (рис.).

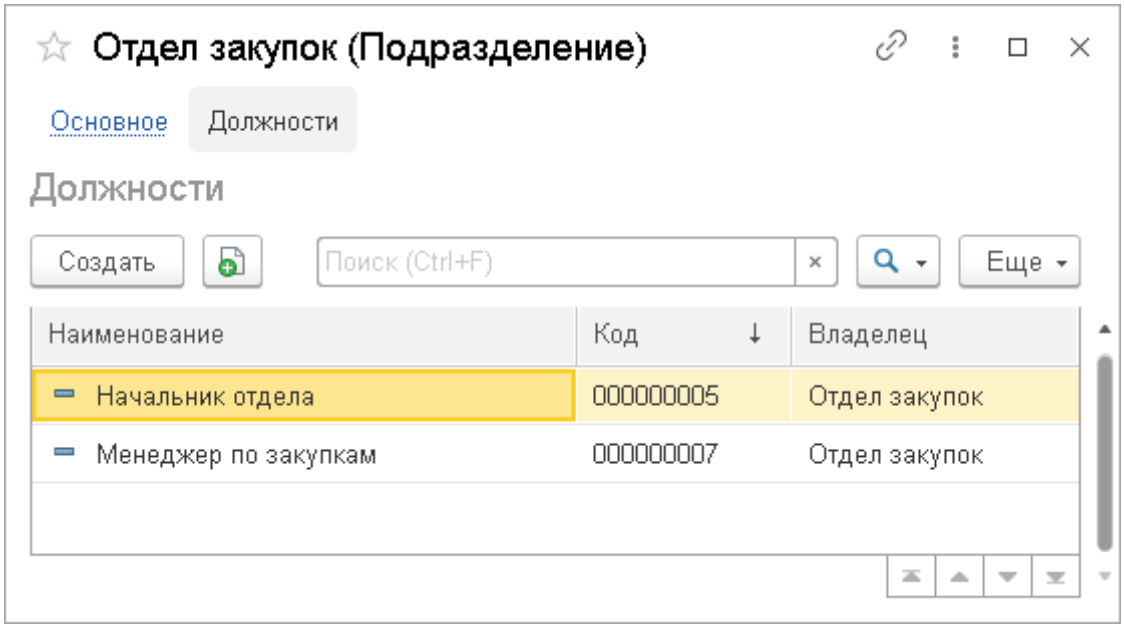


Рис. Список подчиненных элементов на форме создания / редактирования элементов справочника Подразделения

Создание справочника с реквизитами и табличной частью

Теперь создадим справочник Сотрудники. В диалоге создания на вкладке Основное зададим имя Сотрудники, представление объекта Сотрудник. На вкладке Подсистемы отметим Предприятие и Расчет зарплаты.

На вкладке Данные увеличим длину наименования до 50.

Как уже было сказано, у справочника есть стандартный реквизит Наименование, но, поскольку хранится информация о людях, то логично изменить синоним этого реквизита на ФИО. Для этого надо нажать на вкладке Данные кнопку Стандартные реквизиты. В открывшемся списке выберем Наименование и двойным щелчком откроем палитру его свойств. В палитре свойств изменим синоним реквизита на ФИО (рис.).

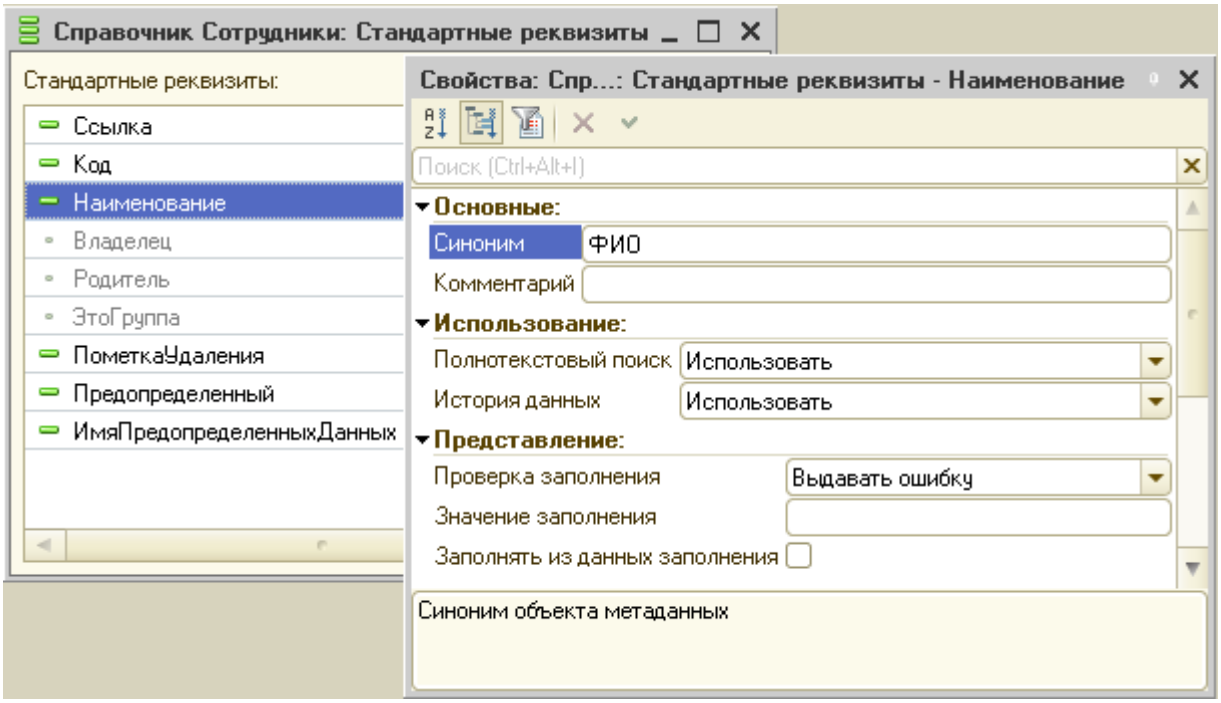


Рис. Определение синонима стандартного реквизита Наименование

Точно так же для стандартного реквизита Код зададим синоним Таб. номер.

Вернемся на вкладку Данные. Для создания нового реквизита справочника надо нажать на пиктограмму над списком реквизитов. Откроется панель свойств нового реквизита, в которой необходимо задать имя и тип реквизита.

Реквизит справочника может иметь стандартный базовый тип (число, строка, дата, булево) или быть ссылкой на другой объект конфигурации.

Создадим следующие реквизиты:

- Подразделение – тип СправочникСсылка.Подразделения;
- Должность – тип СправочникСсылка.Должности;

Создадим табличную часть ТрудоваяДеятельность. Для этого нажмем кнопку Добавить табличную часть ( ) над списком табличных частей справочника. В открывшейся панели свойств зададим имя ТрудоваяДеятельность.

Чтобы добавить реквизит, надо нажать пиктограмму Добавить реквизит ( ) над списком табличных частей справочника.

Добавим следующие реквизиты:

- Организация – тип Строка, длина 100;
- НачалоРаботы – тип Дата, состав даты – Дата;
- ОкончаниеРаботы – тип Дата, состав даты – Дата;
- Должность – тип Строка, длина 100.

Реквизит ОкончаниеРаботы удобно добавить путем копирования элемента НачалоРаботы. Это можно сделать в дереве объектов конфигурации. Раскроем табличную часть ТрудоваяДеятельность, выделим реквизит НачалоРаботы, в меню команд панели дерева объектов нажмем пиктограмму Добавить копированием ( ). Будет создан новый реквизит, которому автоматически присваивается имя НачалоРаботы1 (рис.). Переименовать новый объект можно в панели свойств.

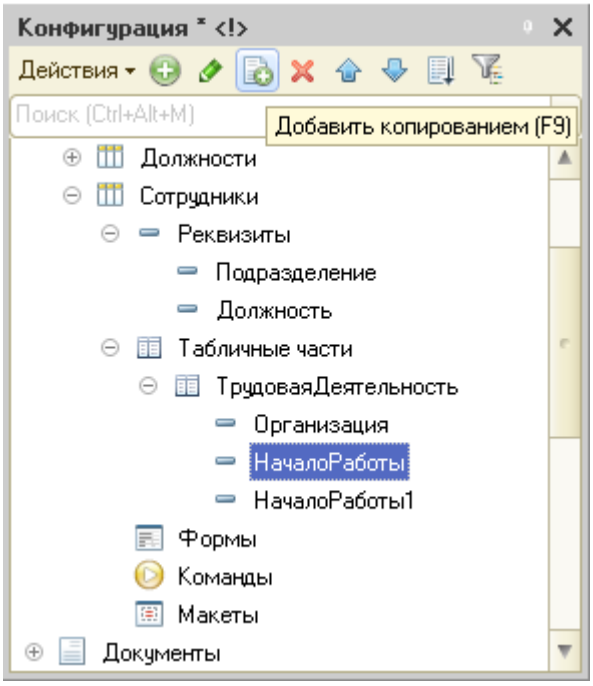


Рис. Добавление реквизита путем копирования

Результат добавления данных в справочник Сотрудники изображен на рис.

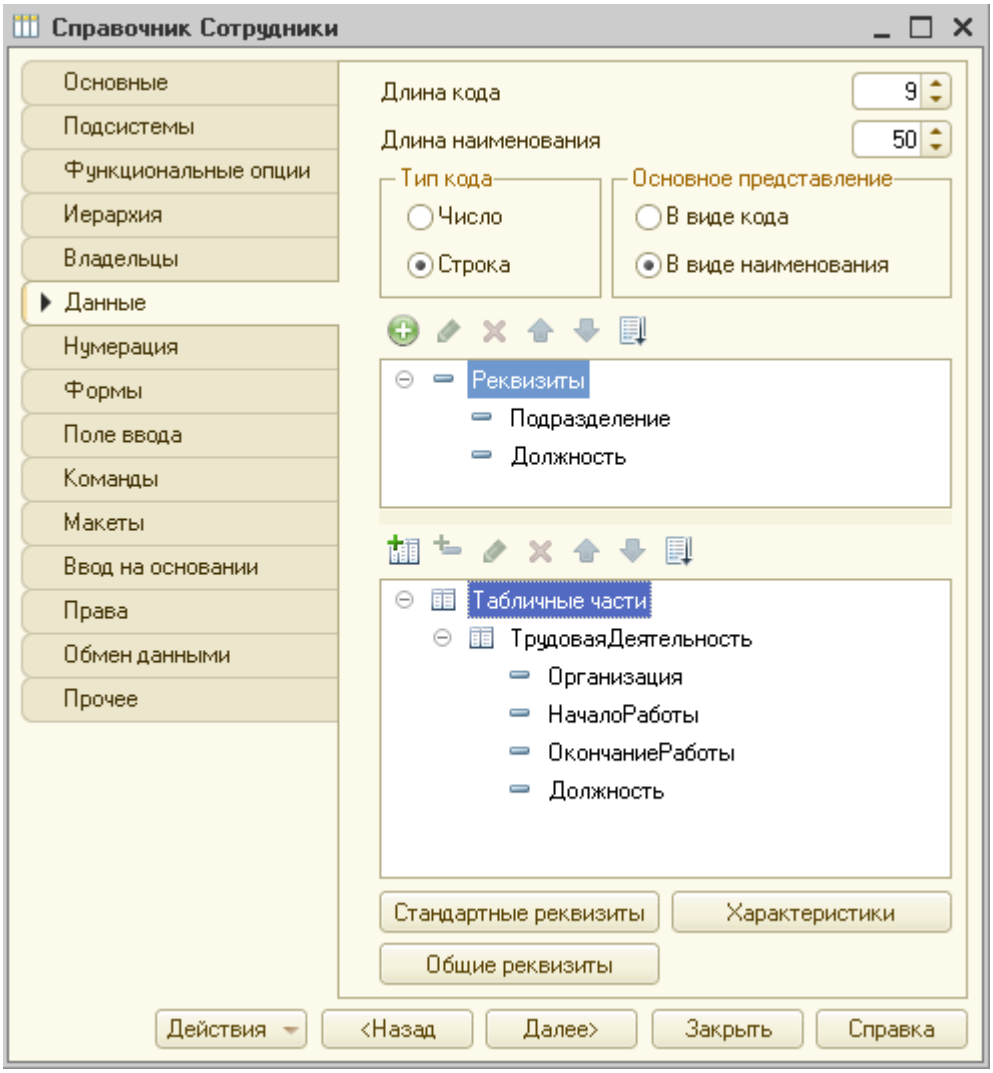


Рис. Создание реквизитов справочника

Связи параметров выбора

Ранее мы создали справочник Подразделения и подчиненный ему справочник Должности. Для того чтобы при выборе подразделения в список должностей загружались только должности одного выбранного подразделения, необходимо настроить для реквизита Должности фильтр, который оставлял бы только нужные значения.

В списке реквизитов справочника Сотрудники выделим реквизит Должности. В панели свойств реквизита найдем свойство Связи параметров выбора и нажмем на пиктограмму команды Выбрать (...) в поле этого свойства. В открывшемся диалоговом окне в списке Доступные реквизиты выберем реквизит Подразделения и добавим его в список параметров, нажав на кнопку Добавить выбранный реквизит в список параметров (>) (рис.).

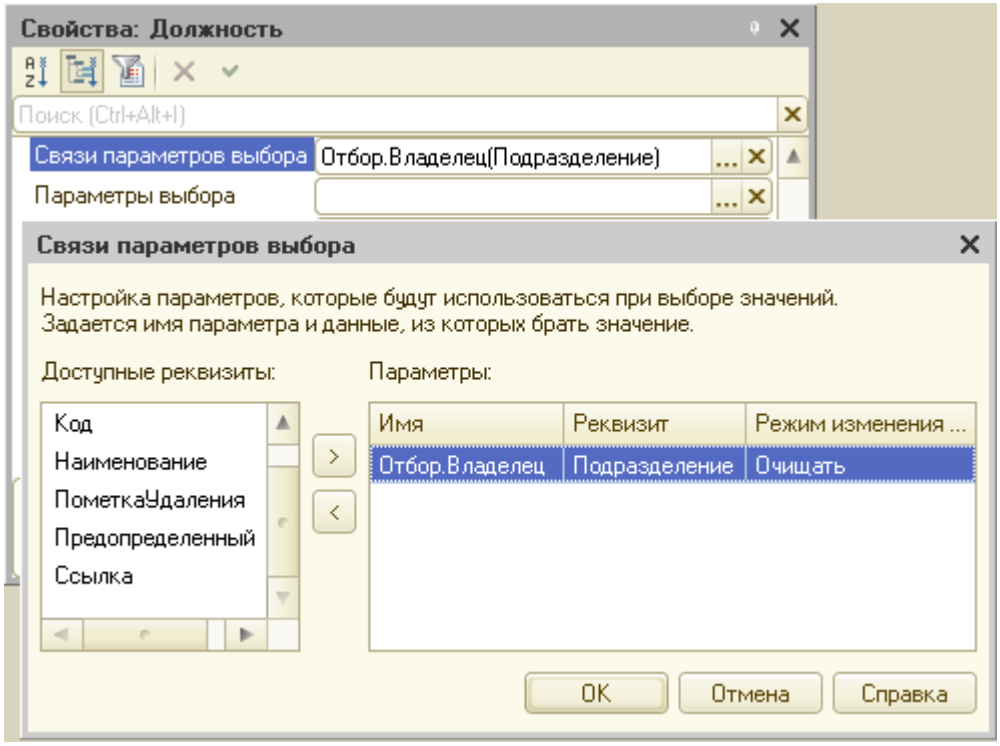


Рис. Определение свойства Связи параметров выбора для реквизита Должности справочника Сотрудники

Созданный таким образом параметр выбора означает, что в списке должностей остаются те элементы, для которых владельцем является значение, выбранное в реквизите Подразделение. В столбце Режим изменения связанного значения указан параметр Очищать. В этом случае при пустом значении реквизита Город список улиц тоже будет очищен.

Создадим форму элемента справочника Сотрудники и расположим основные данные о сотруднике и табличную часть с его трудовой деятельностью на разных вкладках. Запустим приложение в режиме отладки, откроем форму создания нового сотрудника и проверим работу фильтра для поля Должность (рис.).

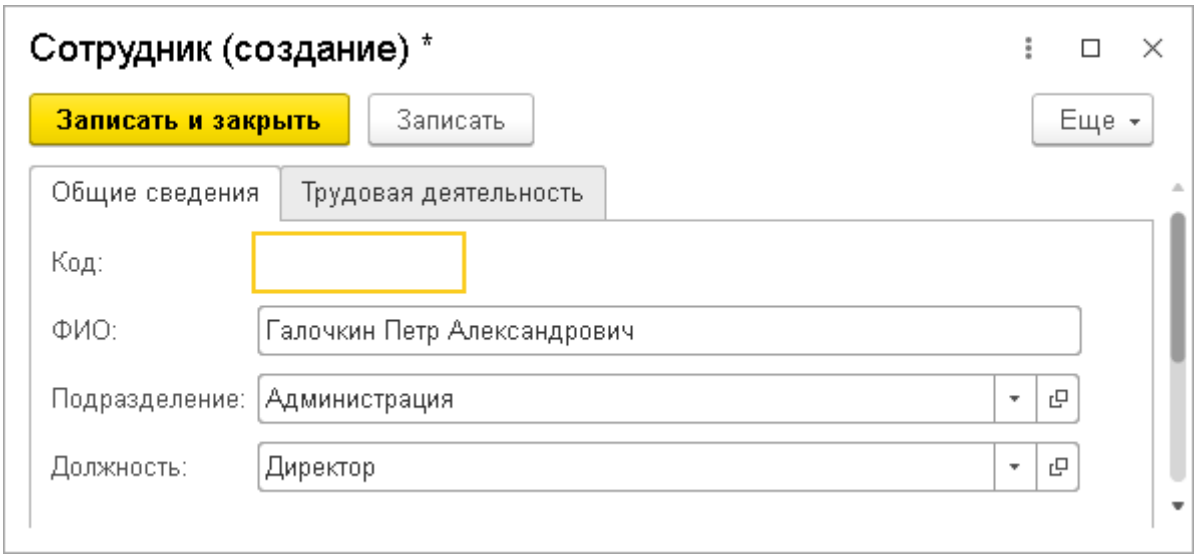


Рис. Форма создания нового сотрудника

Создание справочника с предопределенными элементами

Добавлять элементы в справочник может не только пользователь в процессе работы с готовым приложением, но и разработчик на этапе создания конфигурации. Такие элементы называются *предопределенными*.

Создадим справочник Склады, поместим его в раздел Предприятие. На вкладке Прочее нажмем кнопку Предопределенные и в открывшемся диалоговом окне добавим предопределенный элемент Основной (рис.).

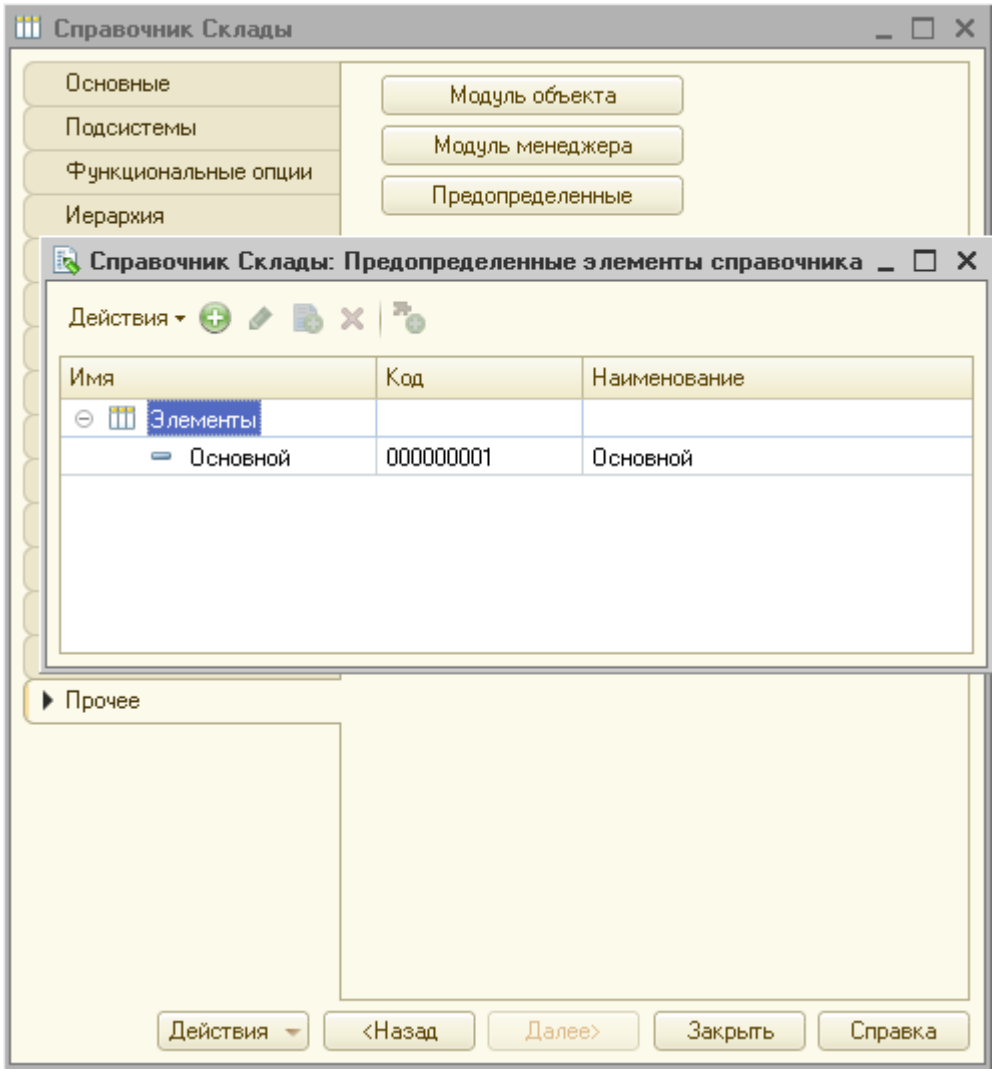


Рис. Создание предопределенных элементов справочника Склады

Запустим приложение и откроем справочник Склады. Как видно, справочник уже содержит предопределенный элемент, отмеченный специальным значком ( ). Добавим в справочник еще один элемент (рис.).

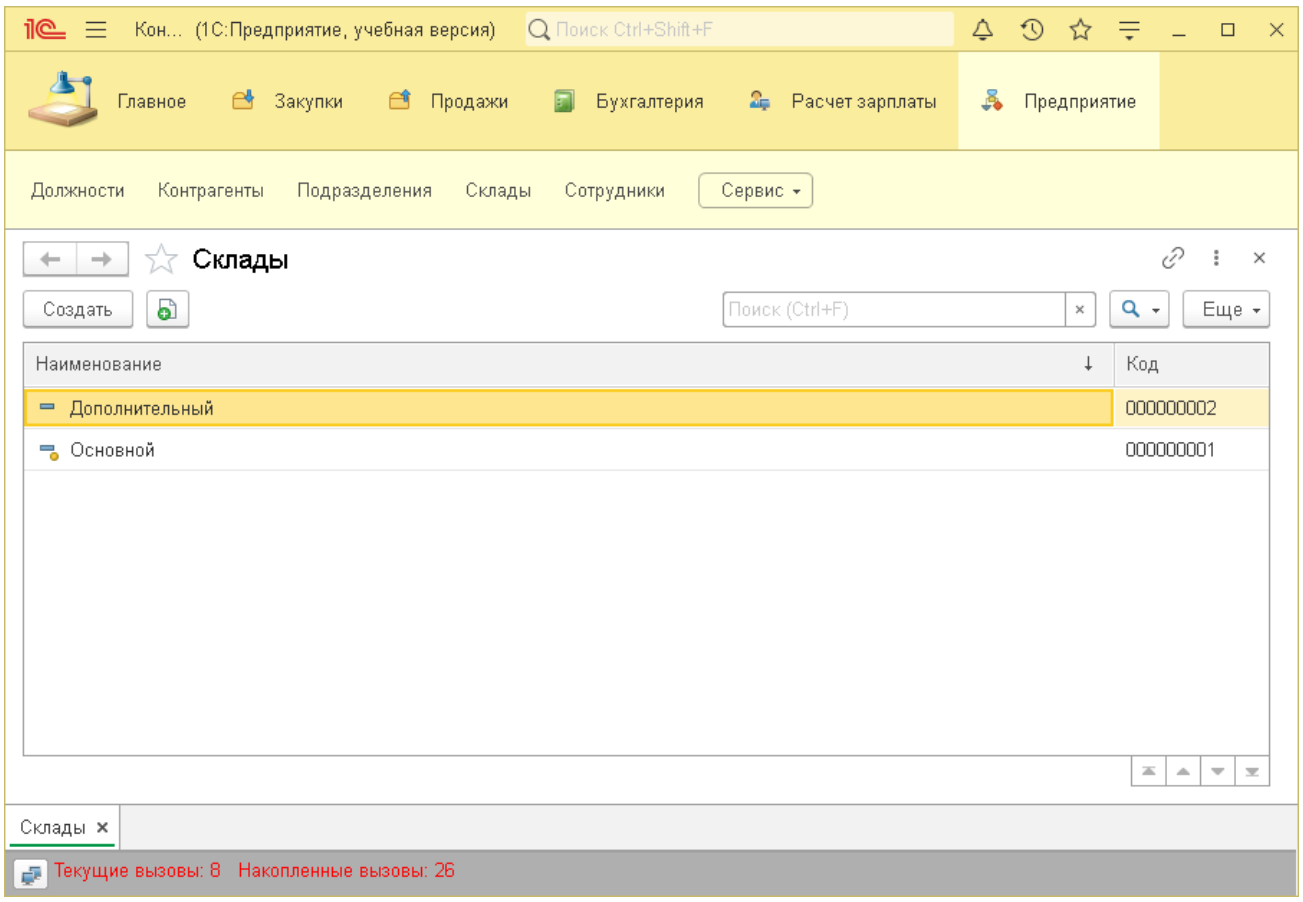


Рис. Справочник Склады с предопределенным элементом

Документы

Документы — это прикладные объекты конфигурации, служащие для регистрации хозяйственных операций, т.е. всех событий, происходящих на предприятии. Примерами хозяйственных операций являются поступление товара на склад, продажа товара, прием сотрудника на работу и т.д.

Каждый документ характеризуется номером, датой и временем. Система поддерживает режим автоматической нумерации документов, при котором она самостоятельно может генерировать номер для нового документа. Кроме этого система позволяет осуществлять контроль уникальности номеров документов, не разрешая создавать документы с одинаковыми номерами.

Важным свойством документа является возможность его *проведения*. При проведении документ изменяет состояние данных в таком объекте конфигурации, как *регистр*.

Система 1С предлагает два режима проведения документа: оперативное и неоперативное.

При *оперативном проведении* документ проводится текущей датой и временем. Как правило, такой режим используется, когда необходимо не просто зафиксировать данные из документа в регистрах, но и получить из них информацию на текущий момент. Например, при продаже товара надо проверить информацию о доступных остатках на складе.

При *неоперативном проведении* документ просто фиксирует свои данные в регистрах системы.

Для документа существуют следующие виды форм:

- форма документа;
- форма списка;
- форма выбора.

Создание документа

Создадим документ ПриходнаяНакладная. На вкладке Основные зададим имя документа и представление списка Приходные накладные (рис.). На вкладке Подсистемы укажем Бухгалтерия и Закупки.

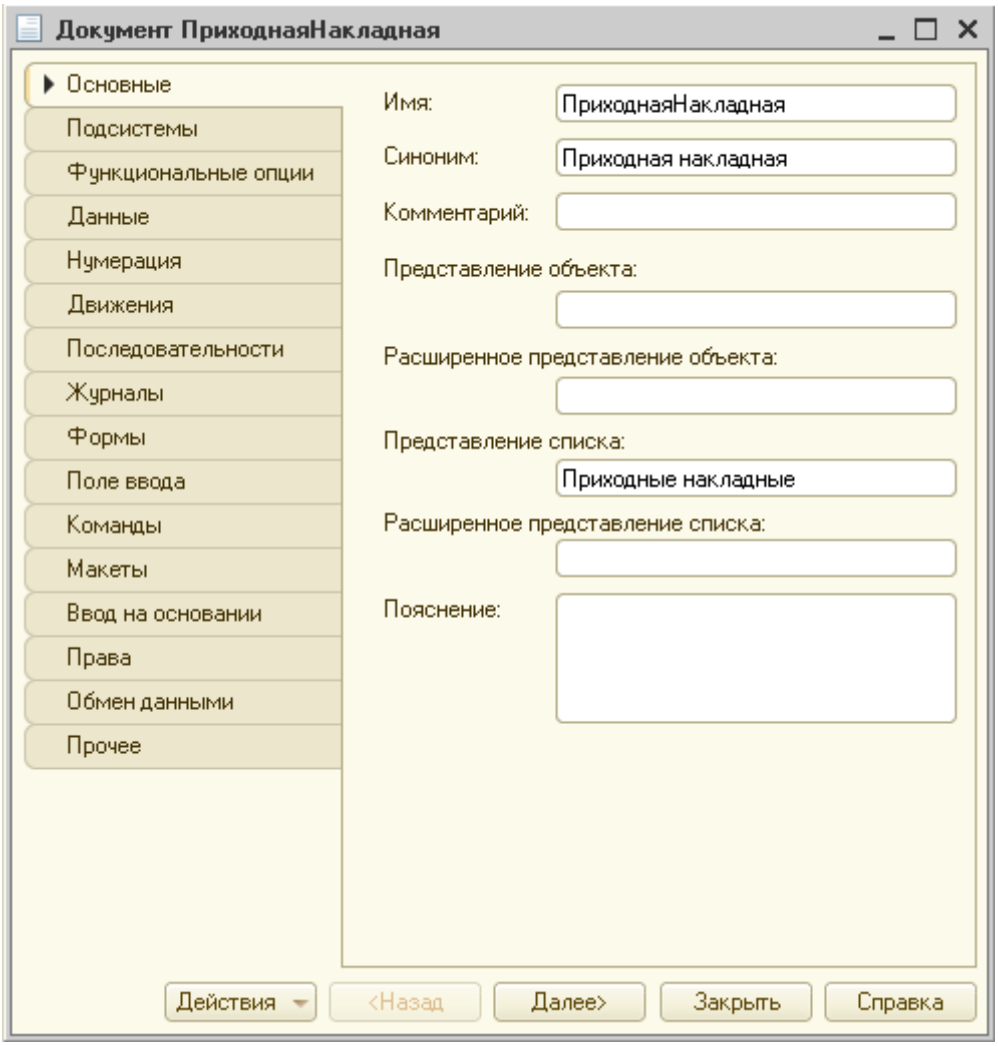


Рис. Диалоговое окно создания документа

На вкладке Данные добавим следующие реквизиты и табличные части (рис.):

- реквизиты:
  - Склад, тип СправочникСсылка.Склады;
  - Поставщик, тип СправочникСсылка. Контрагенты;
- табличная часть Товары, свойство Проверка заполнения – Выдавать ошибку; реквизиты:
  - Товары, тип СправочникСсылка.Номенклатура;
  - Количество, тип Число, длина 10, точность 3, неотрицательное;
  - Цена, тип Число, длина 10, точность 2, неотрицательное;
  - Сумма, тип Число, длина 10, точность 2, неотрицательное.

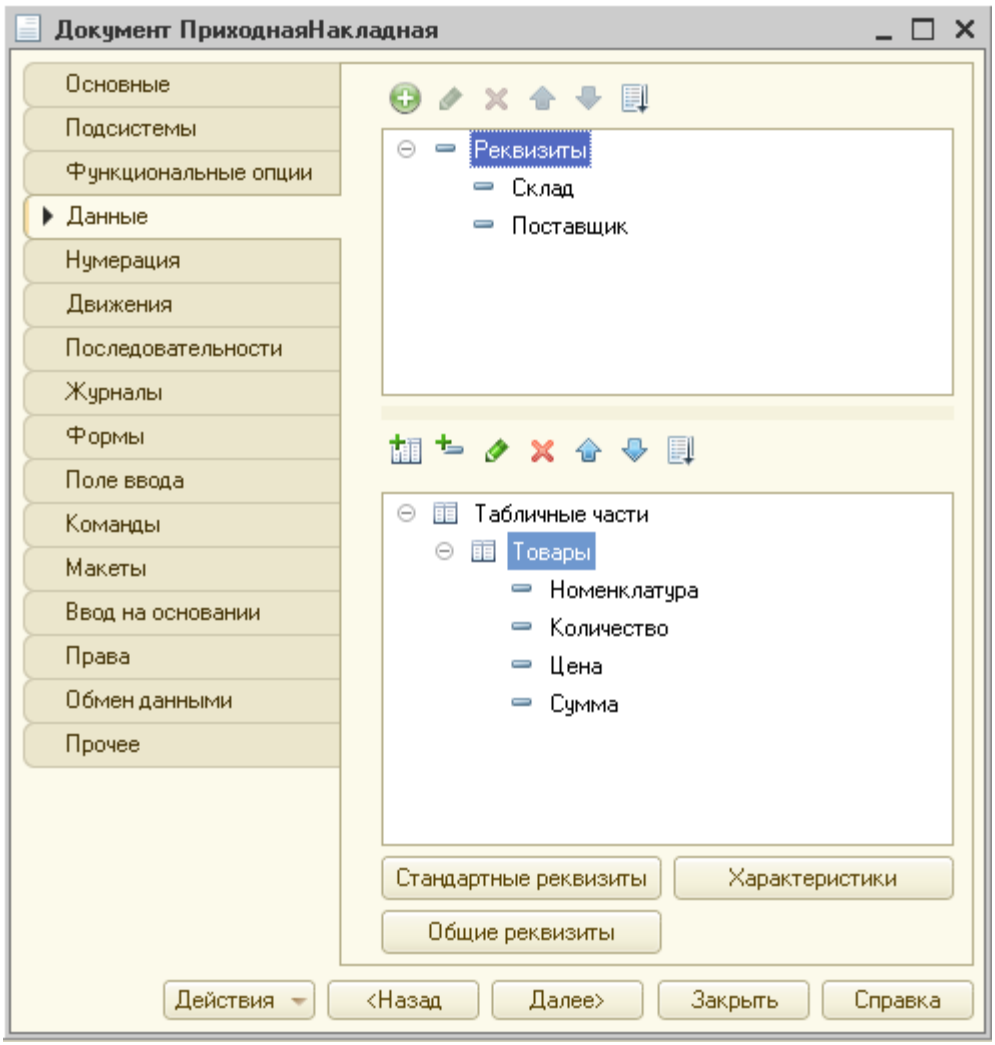


Рис. Вкладка Данные документа Приходная накладная

Установим для свойства Проверка заполнения значение Выдавать ошибку для табличной части и каждого ее реквизита.

Значение заполнения

При создании конфигурации разработчик может присвоить реквизиту значение по умолчанию, которое будет выводиться в поле реквизита при создании объекта конфигурации. За хранение данного значения отвечает свойство Значение заполнения.

В качестве значения заполнения удобно устанавливать предопределенные данные, поскольку они видны в конфигурации.

Предположим, что все закупаемые нашей фирмой товары вначале поступают на склад Основной, поэтому пользователю приложения будет удобно, чтобы при создании приходной накладной в поле Склад сразу было введено значение по умолчанию Основной. Зададим в палитре свойств для реквизита Склад значения заполнения (рис).

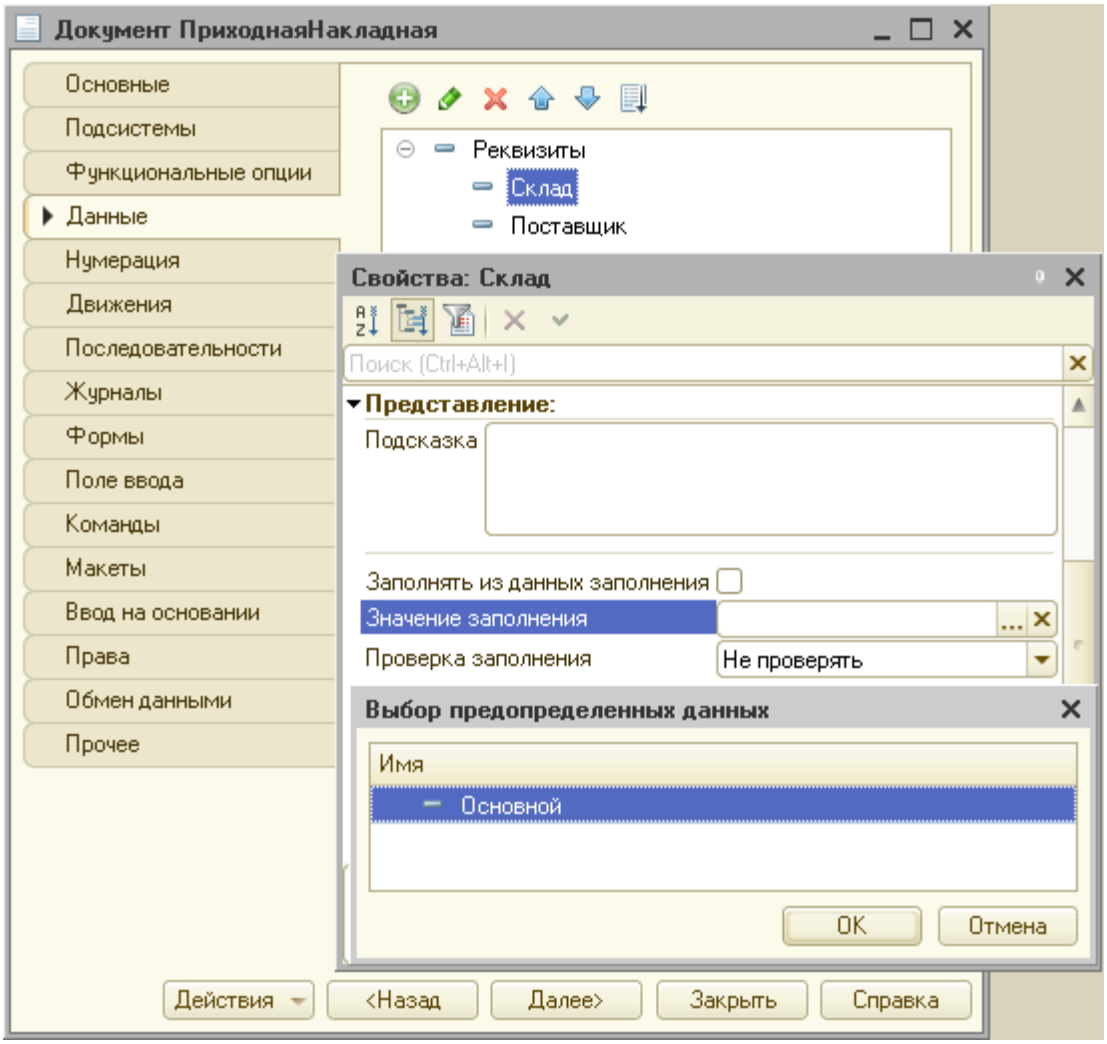


Рис. Определение значения заполнения для реквизита Склад

Запустим приложение и откроем форму ввода документа Приходная накладная (рис.).

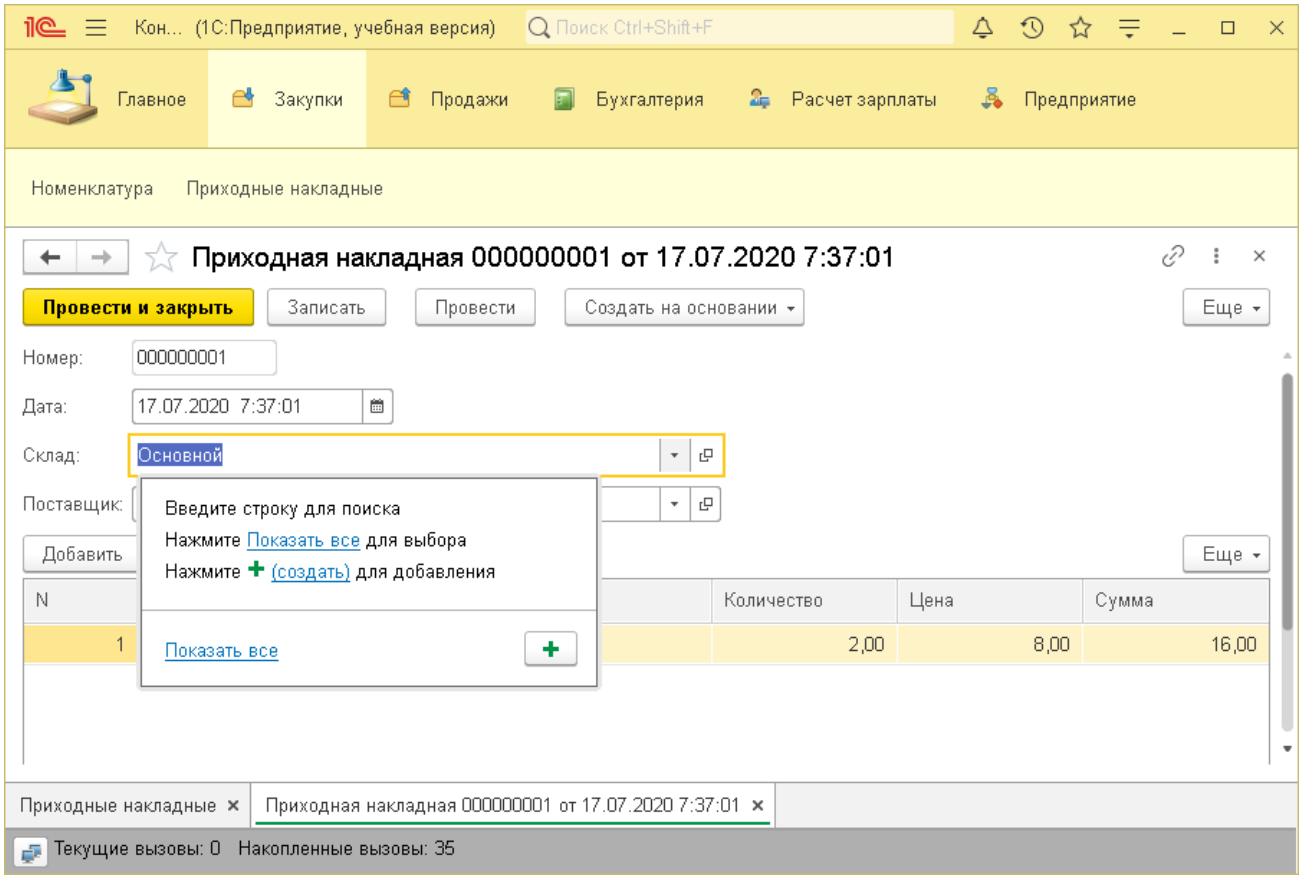


Рис. Форма ввода приходной накладной

Как видно, поле Склад содержит значение по умолчанию Основной. Если нужно его изменить, то необходимо раскрыть выпадающий список.

Быстрый выбор

По умолчанию при выборе элементов справочника в выпадающем списке предлагается открыть отдельную форму выбора, нажав на ссылку Показать все. Однако, если заранее известно, что справочник содержит небольшое количество элементов, то удобнее будет сразу загрузить все элементы в список. Для этого необходимо установить для справочника свойство Быстрый выбор.

Откроем редактор справочника Склады на вкладке Поле ввода и установим свойство Быстрый выбор (рис.).

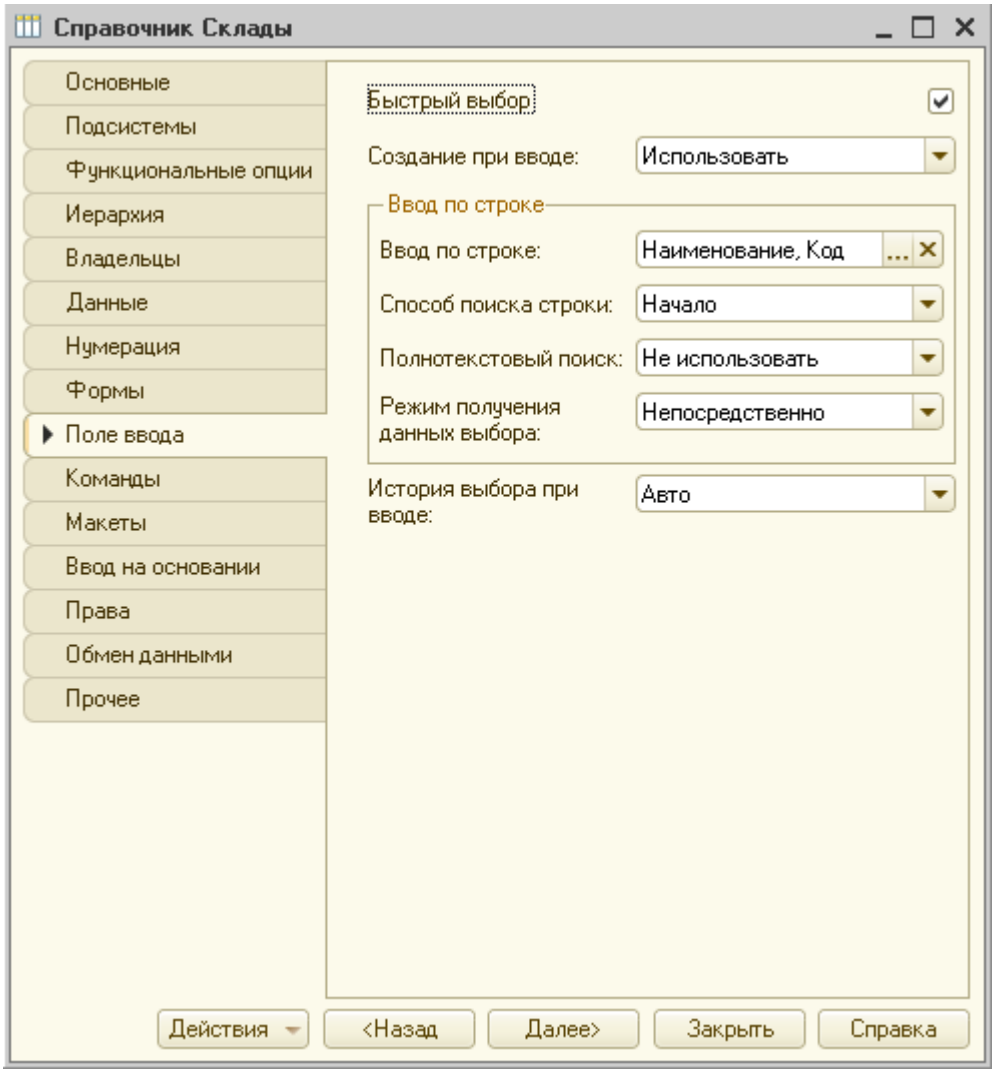


Рис. Установка свойства Быстрый выбор для справочника Склады

Запустим приложение и снова откроем форму создания приходной накладной (рис.).

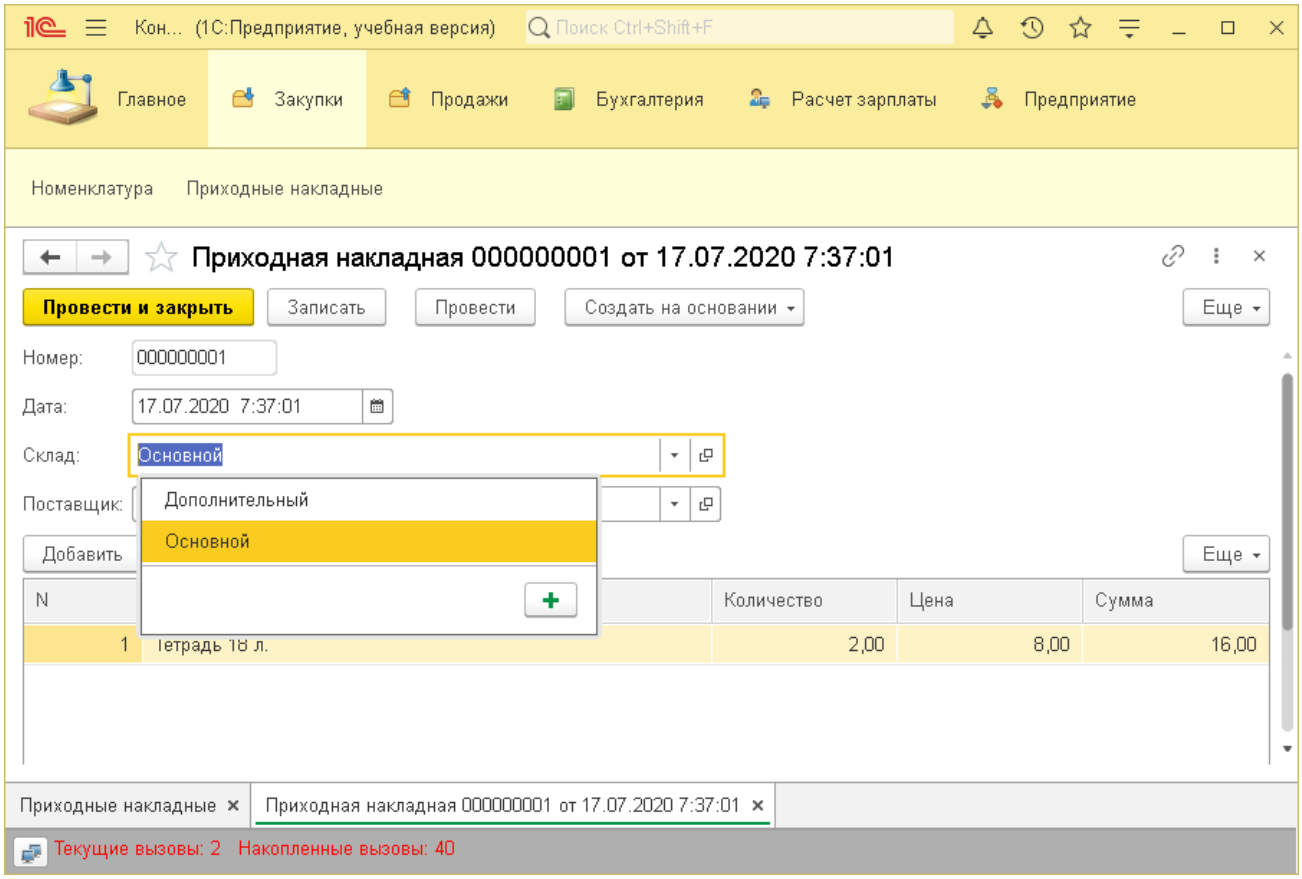


Рис. Список выбора значений после установки свойства Быстрый выбор

Вычисление итогов для табличной части

По умолчанию все колонки табличной части имеют заголовки, однако есть возможность отобразить для них *подвал* – специальную область в нижней части таблицы, в которую можно выводить, например, сумму числовых данных столбца.

Стандартная форма документа не предусматривает настройку таблицы, поэтому необходимо создать новую форму документа. В конструкторе формы надо выделить табличную часть Товары и в панели свойств установить флажок Подвал (рис.).

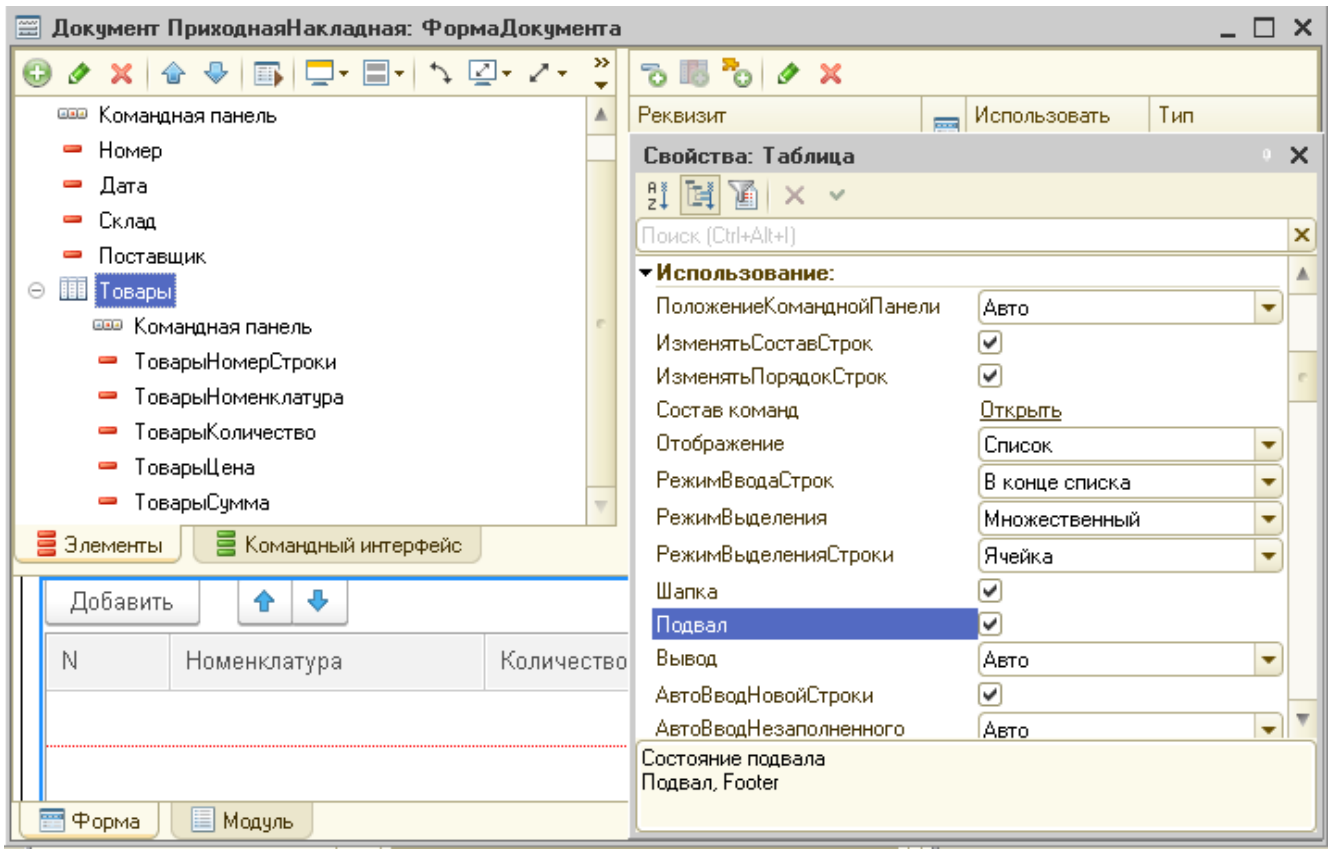


Рис. Выбор отображения подвала для табличной части документа

Далее необходимо выделить реквизит табличной части ТоварыКоличество, в панели свойств задать свойства Текст подвала и Путь к данным подвала (рис.).

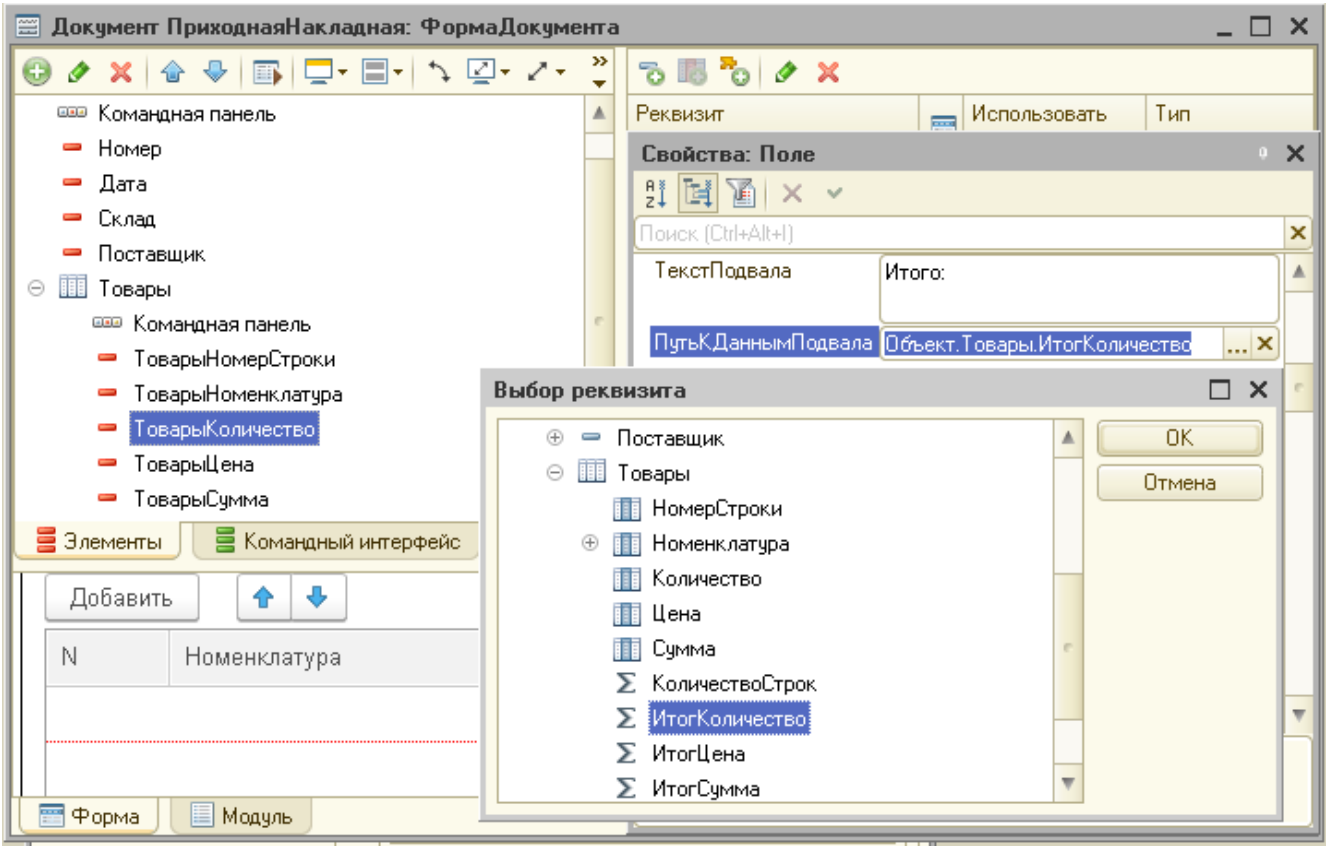


Рис. Выбор реквизита для отображения в подвале столбца табличной части

Таким же образом установим отображение итога для поля Сумма.

Добавление приходных накладных

Запустим конфигурацию в режиме отладки и откроем форму создания новой накладной (рис.).

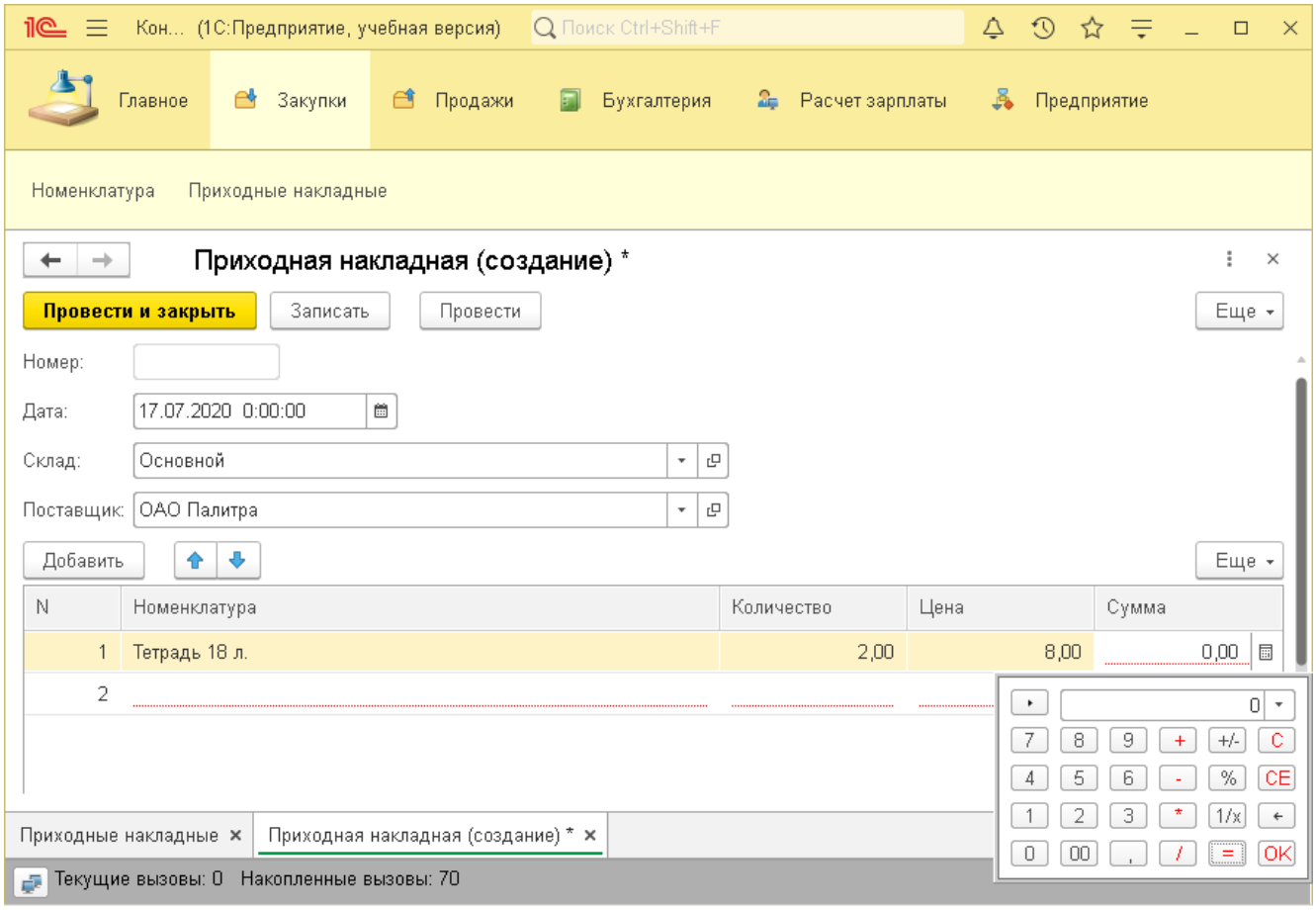


Рис. Форма создания новой накладной

Как видно, в форме поле Номер не заполнено, но он будет сгенерирован автоматически. В поле Дата система автоматически подставила текущую дату и нулевое время, т.к. документ еще не проведен. В качестве времени документа при оперативном проведении ему присваивается оперативная отметка времени. Поле Склад содержит значение по умолчанию Основной, как было задано в свойстве Значение заполнения. Все поля табличной части содержат красное подчеркивание, т.к. для ее реквизитов было установлено свойство Проверка заполнения.

Для удобства пользователя из числовых полей документа можно вызвать встроенный калькулятор. После выполнения расчетов и нажатия кнопки ОК или клавиши Enter значение сохраняется в ячейке.

Создание документа Расходная накладная

Создадим документ Расходная накладная копированием документа Приходная накладная. Появится документ ПриходнаяНакладная1. Переименуем его в РасходнаяНакладная, изменим поле Представление объекта на Расходные накладные, на вкладке Подсистемы отметим Продажа товаров и Бухгалтерия. На вкладке Данные переименуем реквизит Поставщик в Покупатель и добавим реквизит Сотрудник (рис.).

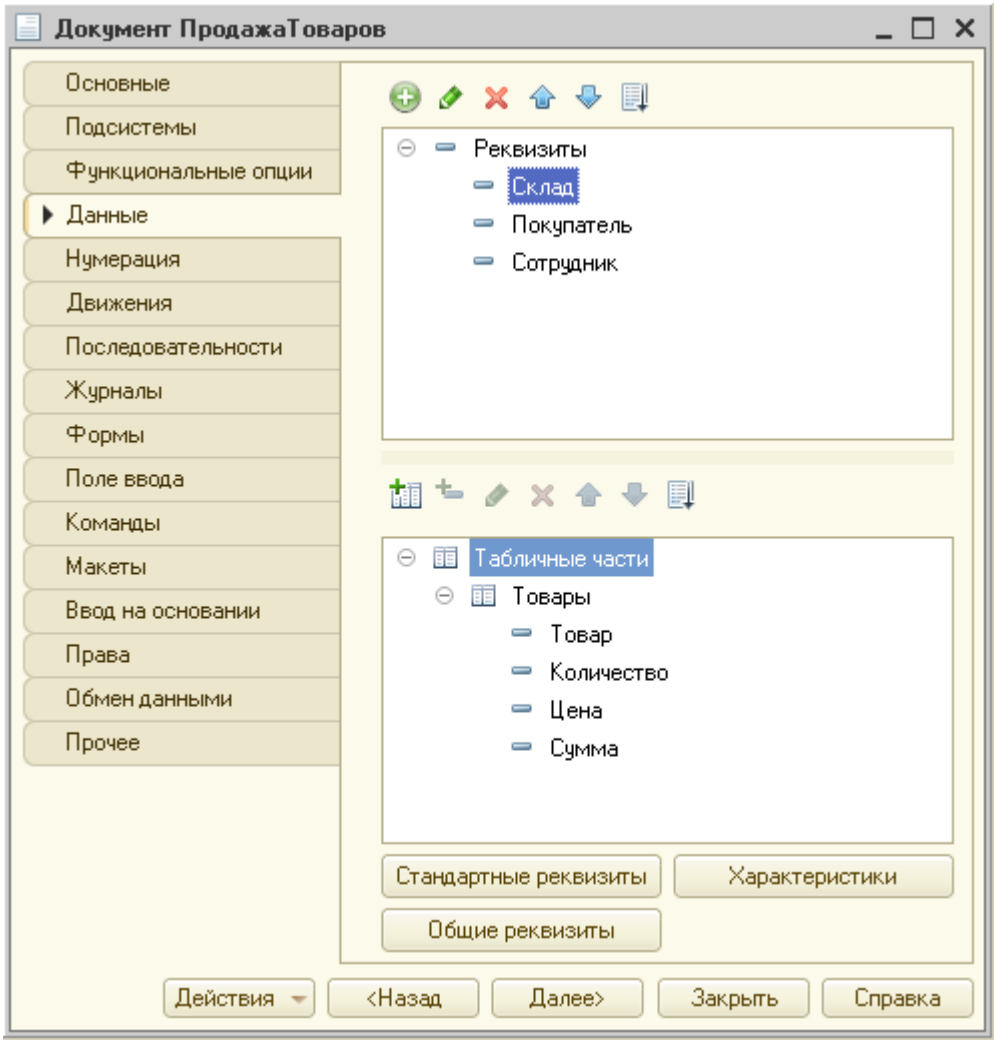


Рис. Документ Продажа товаров

Запустим приложение в режиме отладки и создадим несколько документов Приходная накладная (рис.).

Обратим внимание, что в пользовательском режиме на форме списка также есть кнопка Создать новый элемент копированием текущего (), при нажатии на которую система скопирует выделенный документ и откроет его для редактирования.

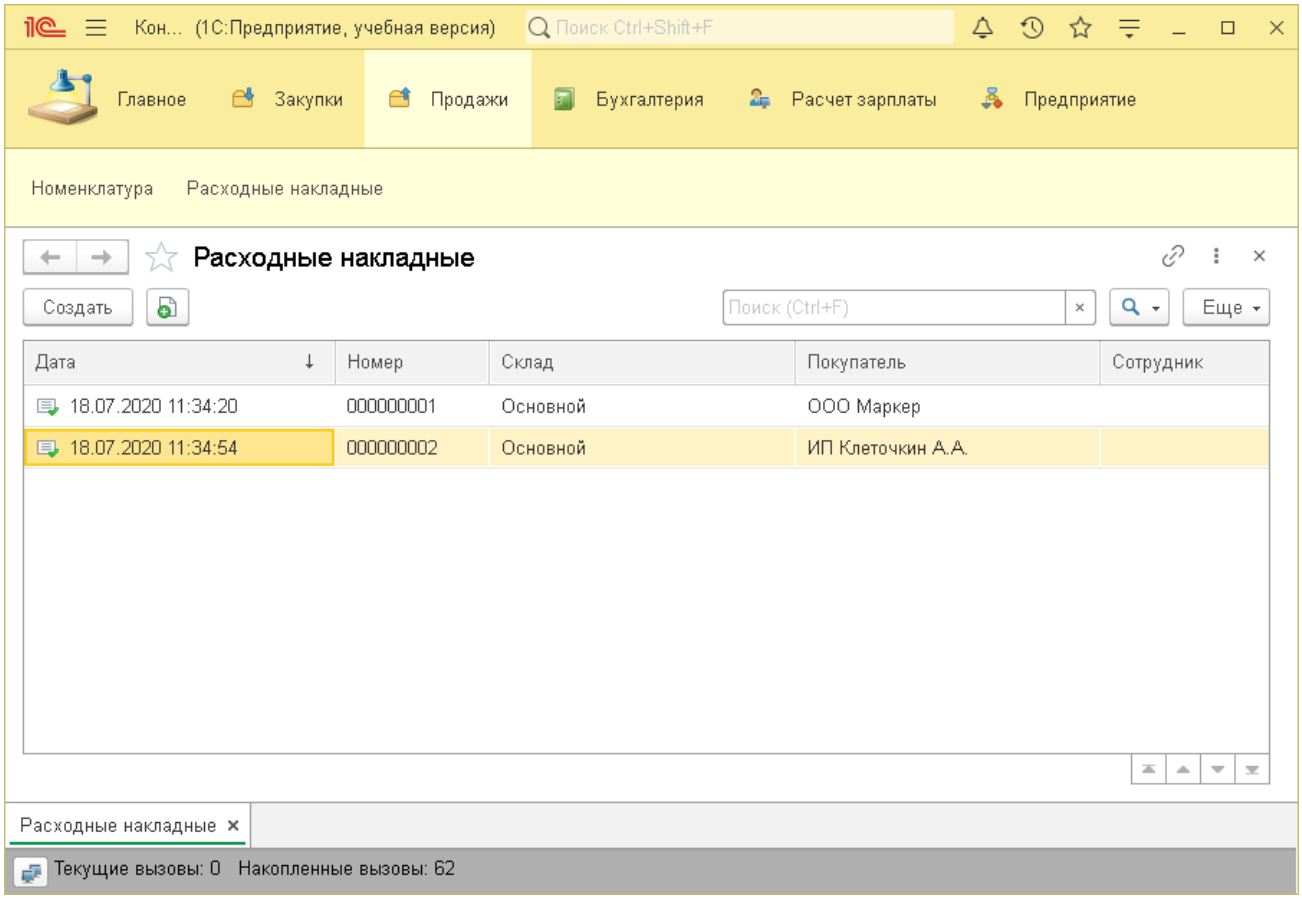


Рис. Список приходных накладных

Ввод нового документа на основании существующего

Система 1С предоставляет пользователю возможность создавать новые документы не только путем редактирования копии документа такого же вида, но и использовать в качестве исходного копию документа другого вида. Данный механизм называется *Ввод на основании*.

Откроем в конфигураторе вкладку Ввод на основании редактора документа Расходная накладная и в список Вводится на основании добавим документ Приходная накладная (рис.).

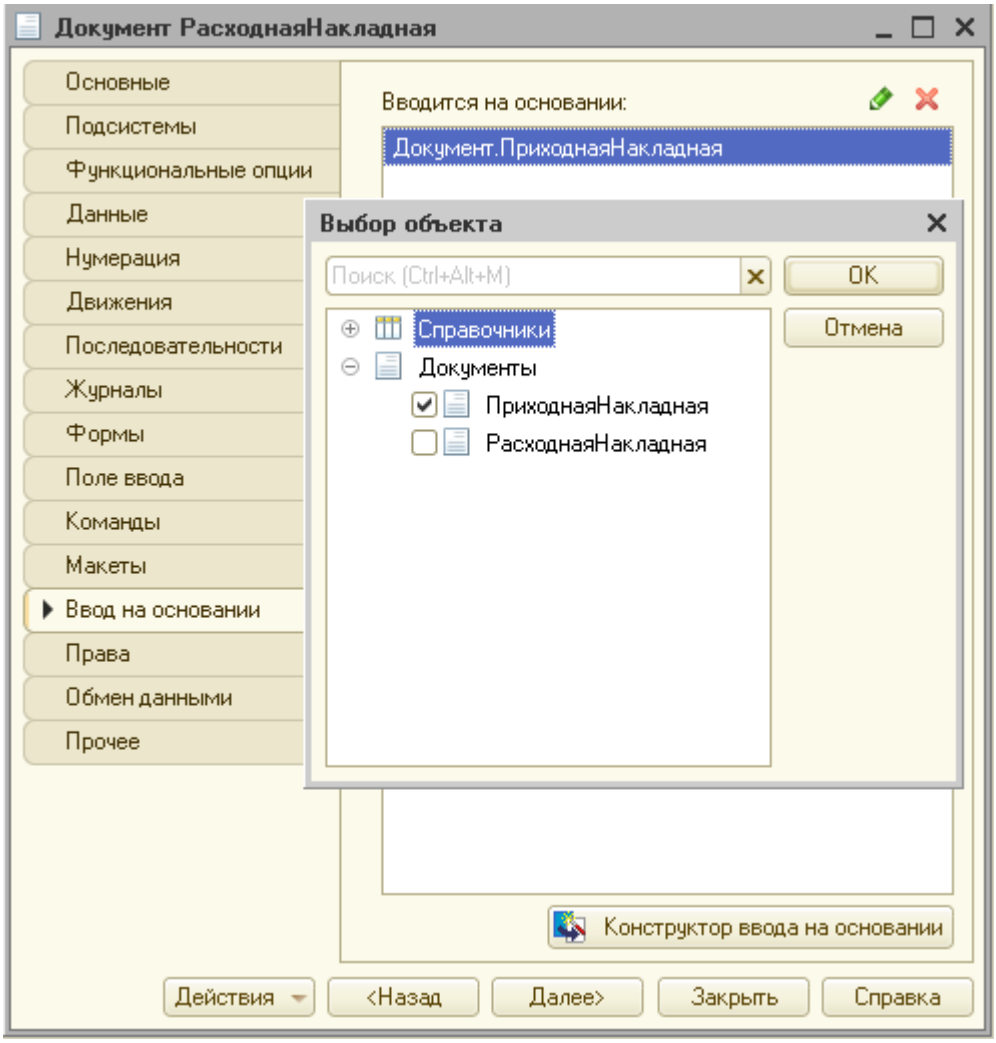


Рис. Выбор документа для ввода на основании

Далее необходимо нажать кнопку Конструктор ввода на основании. Поскольку имена практических всех реквизитов в обоих документах совпадают, то в форме конструктора можно просто нажать кнопку Заполнить выражения и система автоматически заполнит совпадающие поля (рис.).

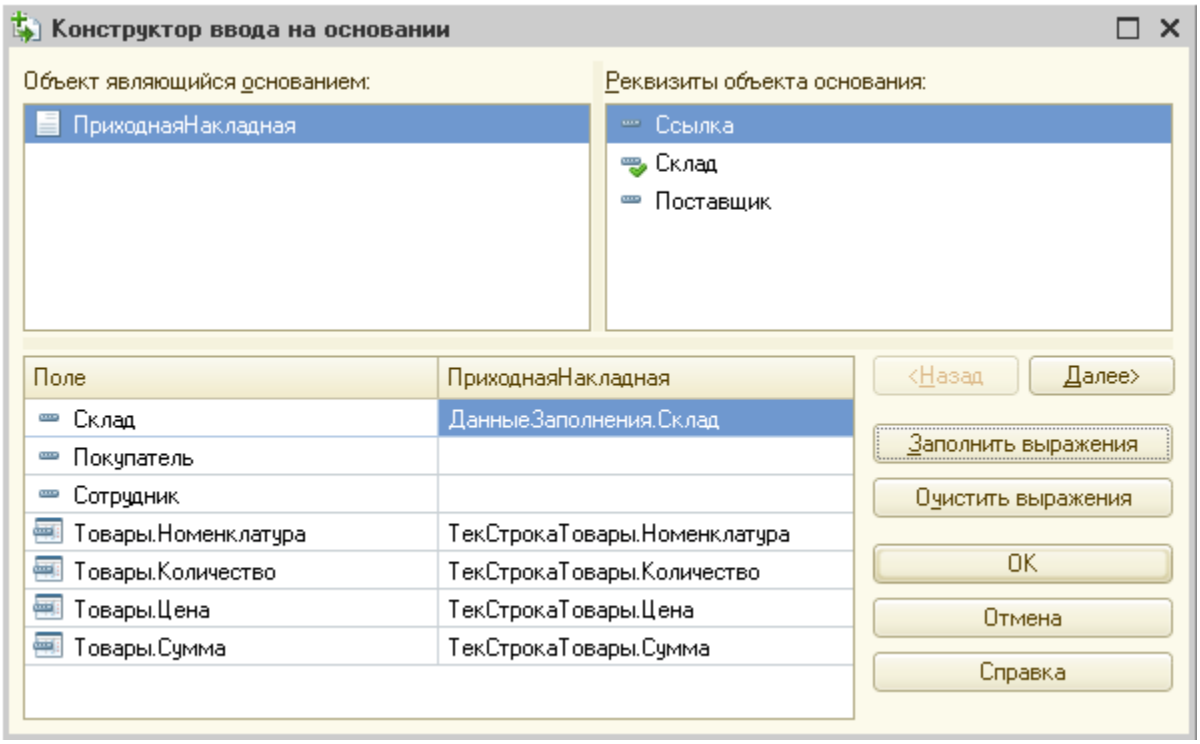


Рис. Конструктор ввода на основании

После нажатия на кнопку ОК откроется модуль объекта с автоматически сформированной процедурой Обработка заполнения (рис.).

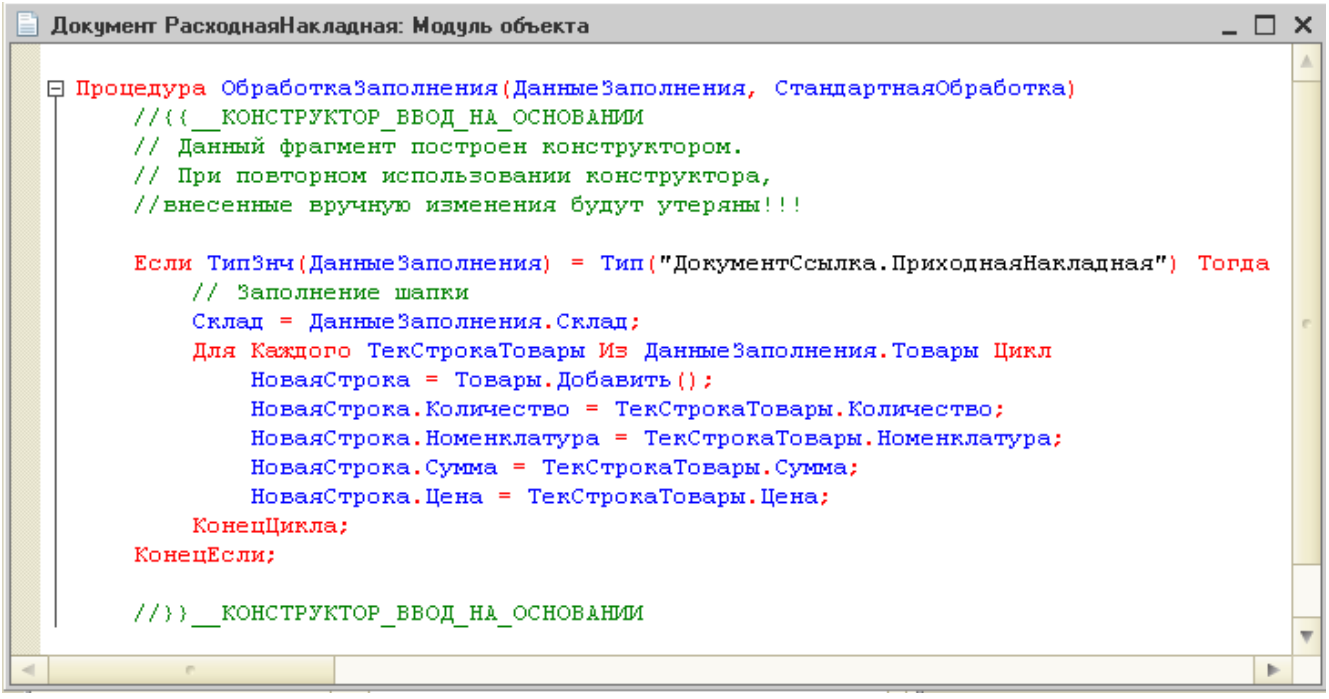


Рис. Процедура ОбработкаЗаполнения()

После добавления механизма ввода на основании в панели инструментов в форме списка приходных накладных появится подменю Создать на основании (рис.).

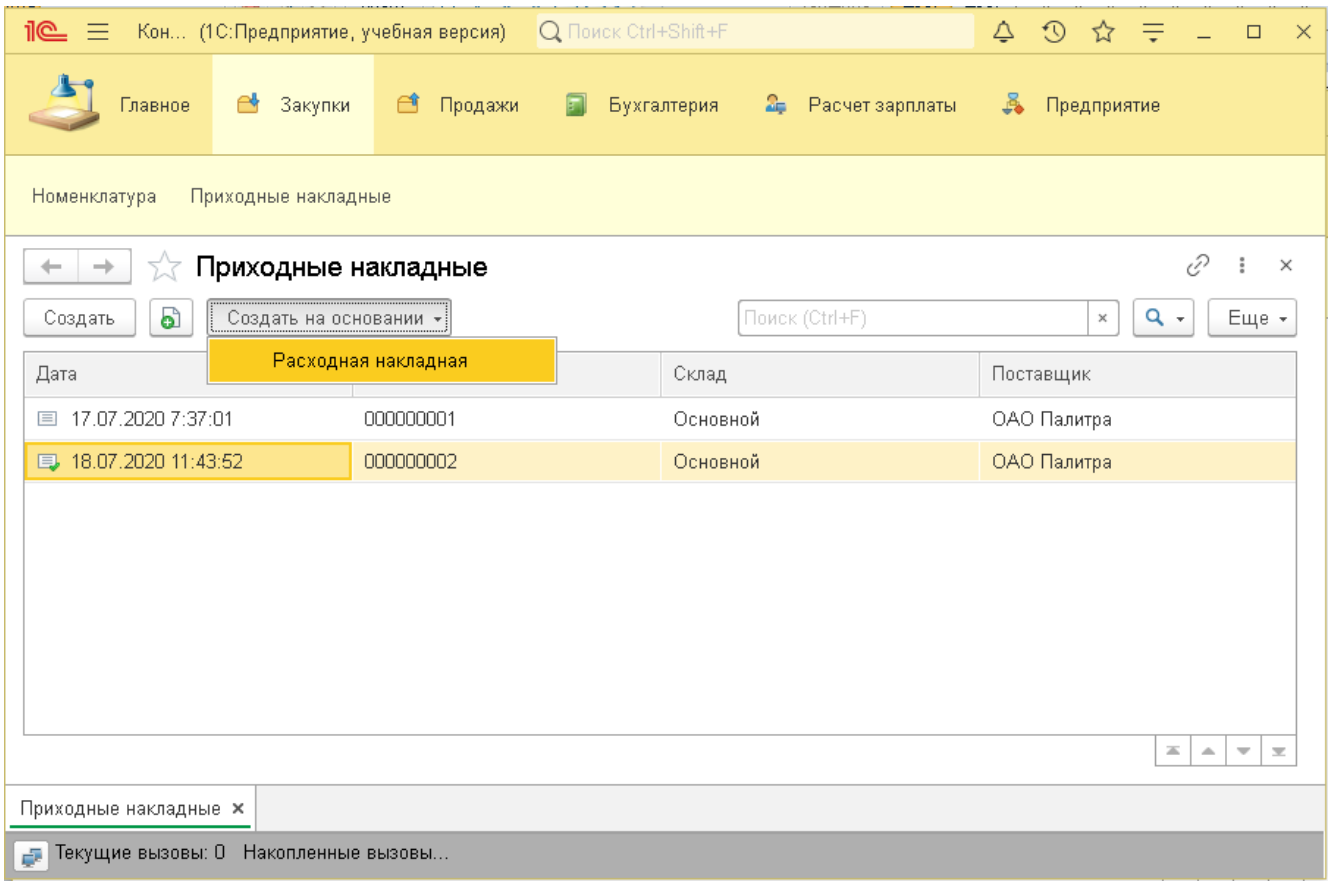


Рис. Добавление кнопки Создать на основании

Журнал документов

Журналы документов — это прикладные объекты конфигурации, которые могут содержать документы различных видов. Обычно в журнал объединяются документы, схожие по назначению, например, складские, финансовые, банковские и т.д. Для журнала документов могут быть определены графы, предназначенные для отображения реквизитов документов разного вида, отнесенных к данному журналу:

Журнал может содержать произвольное количество видов документов. В прикладном решении может быть создано произвольное количество журналов документов.

Создадим журнал документов Складские документы и добавим его в подсистемы Предприятие и Бухгалтерия. На вкладке Данные выберем документы Приходная накладная и Расходная накладная (рис.) и добавим графы журнала:

- Склад – Ссылки: реквизит Склад у приходной и расходной накладных;
- Контрагент – Ссылки: реквизиты Поставщик и Покупатель (рис.).

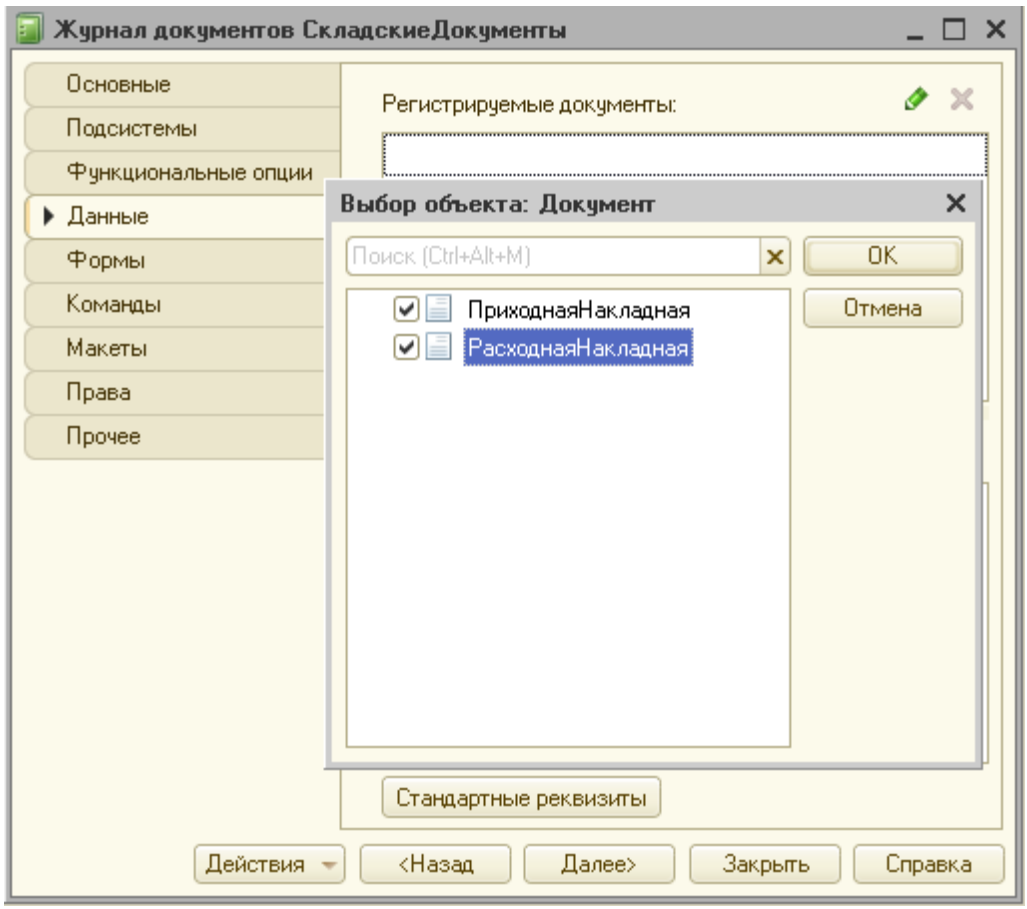


Рис. Выбор документов для журнала Складские документы

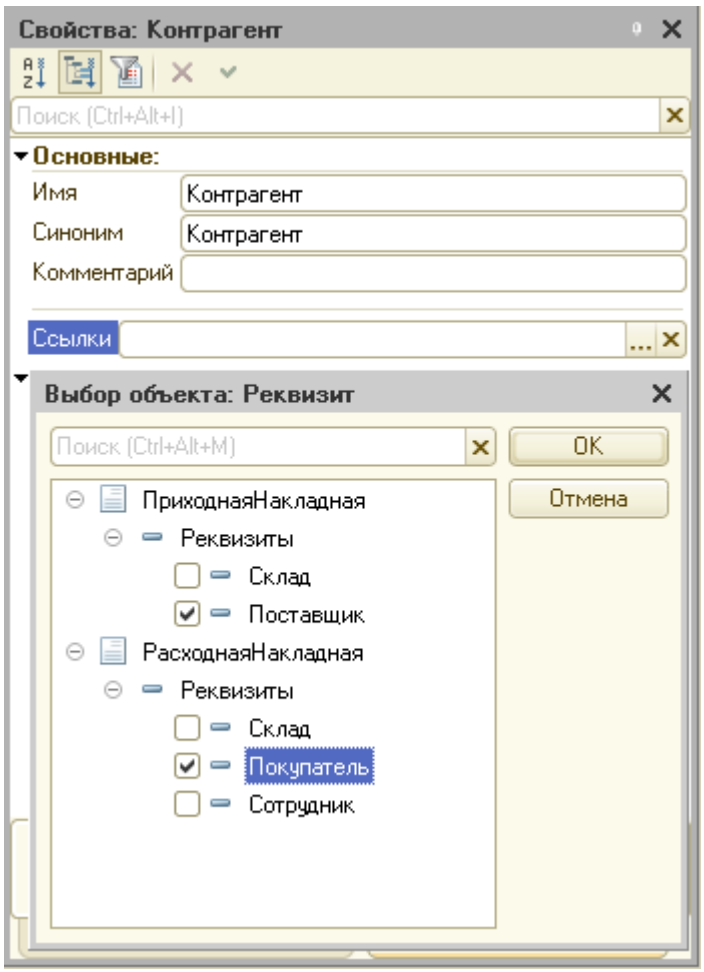


Рис. Выбор реквизитов документов для отображения в журнале

Добавление команды создания новых элементов на панель команд подсистемы

При добавлении объекта конфигурации в подсистему в панели команд автоматически появляется команда открытия формы списка элементов. Для создания нового элемента необходимо перейти на форму списка и уже там нажать на кнопку Создать. Однако если для каких-либо объектов новые элементы нужно создавать достаточно часто (например, документы накладных), то система предоставляет возможность добавления кнопки создания прямо в панель команд подсистемы.

Это можно сделать с помощью редакторов командного интерфейса. Например, откроем редактор Все подсистемы. В разделе Закупки в списке команд установим видимость для команды Приходная накладная.Создать (рис.).

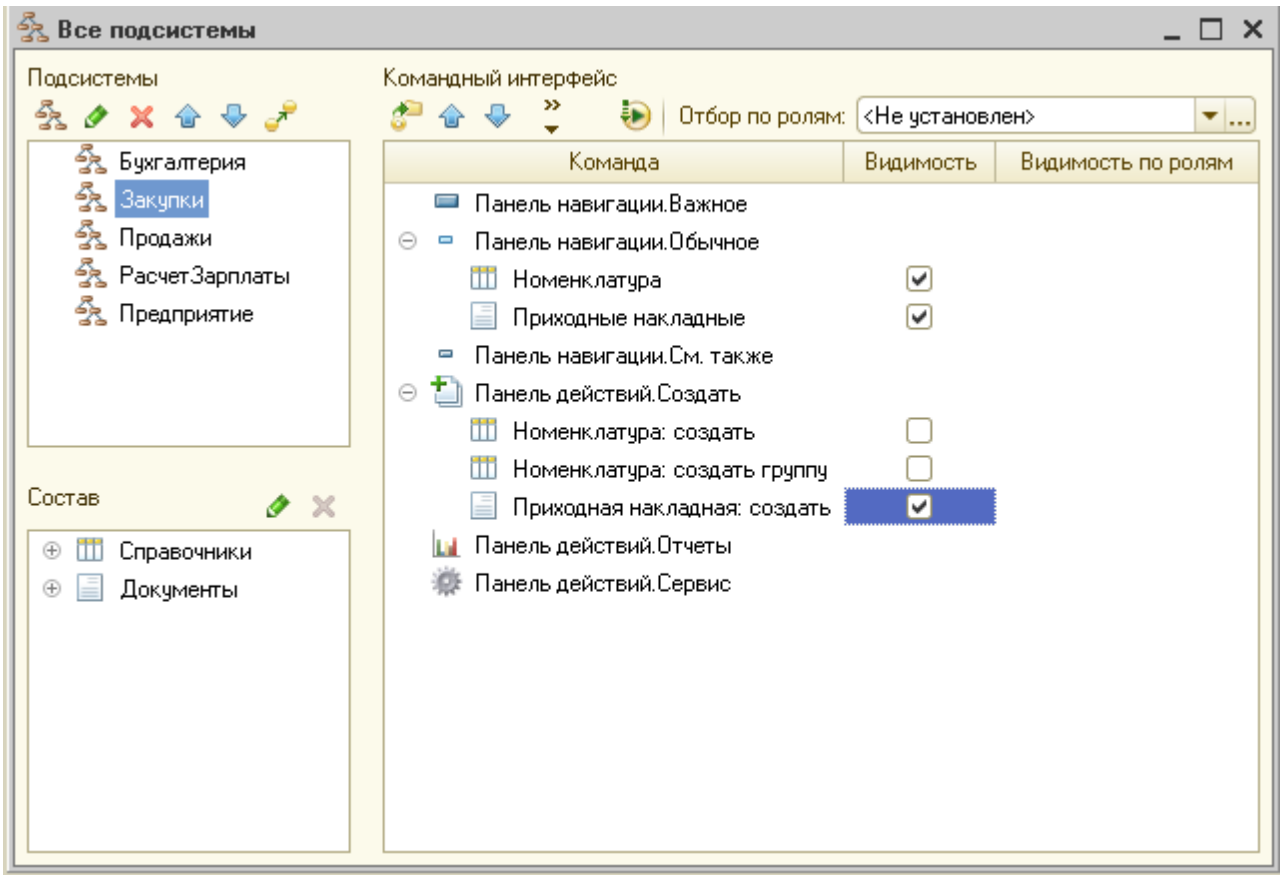


Рис. Установка видимости для команды создания приходных накладных

Запустим приложение и откроем подсистему Закупки (рис.).

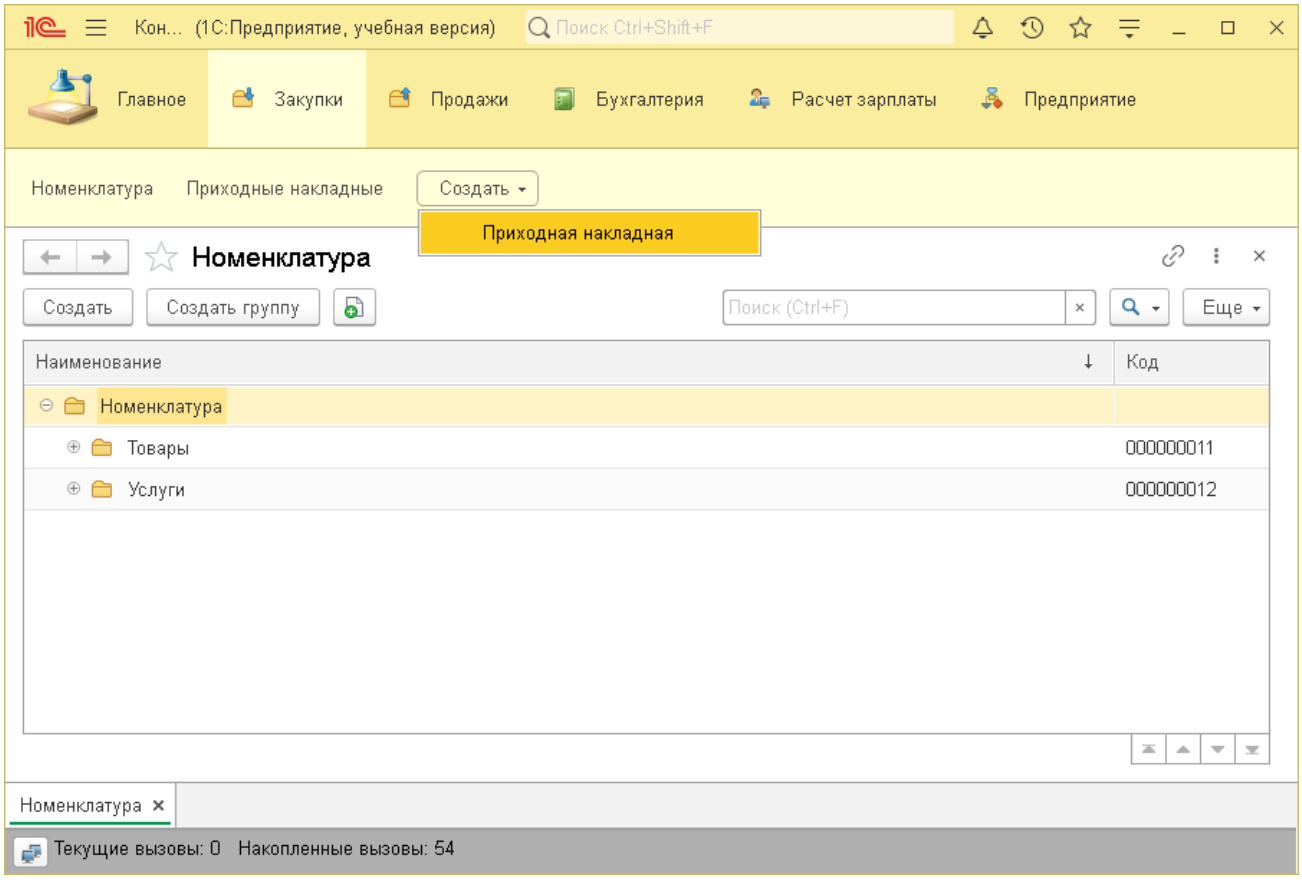


Рис. Результат добавления команды создания новой приходной накладной на панель команд раздела Закупки

Задание на лабораторную работу

1. Выполнить все примеры, рассмотренные в лабораторной работе.
2. Найти в открытых источниках описание типовых конфигураций 1С (1С: ERP, 1С: Бухгалтерия и т.д.). Сохранить скриншоты форм со сведениями о контрагентах и о сотрудниках из типовой конфигурации.
3. По примеру форм из типовой конфигурации добавить еще минимум по 5 реквизитов в справочники Контрагенты и Сотрудники. Использовать реквизиты разных типов (примитивные, ссылки на перечисления и другие справочники).
4. Добавить реквизиты на формы элементов справочников Контрагенты и Сотрудники.
5. Выгрузить информационную базу и сохранить ее для следующей лабораторной работы.

Лабораторная работа №5-6
Основы программирования в системе 1С: Предприятие 8.3

Особенности встроенного языка программирования 1С

Встроенный язык является важной частью технологической платформы «1С: Предприятия 8» и позволяет разработчику описывать собственные алгоритмы функционирования прикладного решения.

Встроенный язык имеет много общих черт с другими языками, такими как C++, Java, PHP, что облегчает его освоение начинающими разработчиками. Однако он не является прямым аналогом какого-либо из перечисленных языков.

Согласно идеологии системы 1С: Предприятие большая часть прикладного решения создается разработчиком путем визуального конструирования — создания новых объектов конфигурации, задания их свойств, форм представления, взаимосвязей и т.д., а встроенный язык используется лишь для того, чтобы определить поведение объектов прикладного решения, отличное от типового, и создать собственные алгоритмы обработки данных.

К особенностям встроенного языка платформы 1С относятся:

- поддержка двуязычного синтаксиса – система позволяет разработчику при написании текста на встроенном языке использовать как русскоязычные операторы, так и их аналоги на английском языке (например, Выбрать и Select);
- мягкая типизация — тип переменной не указывается прямо, а определяется типом значения, которое она содержит, и может изменяться в процессе работы;
- событийная зависимость – фрагменты текста на встроенном языке вызываются только при возникновении определенных заранее известных событий, например, добавления или редактирования элементов.

Обозначения стандартных элементов программного кода

Обозначения стандартных операций и некоторых других элементов программного кода на встроенном языке программирования 1С приведены в табл.

№	Элементы программного кода	Обозначение
1	Символ конца строки программного кода	;
2	Оператор присваивания	=
3	Арифметические операции (сложение, вычитание, умножение, деление, остаток от деления)	+, -, *, /, %
4	Логические операции (конъюнкция, дизъюнкция, логическое отрицание)	И, ИЛИ, НЕ
5	Операции сравнения (равно, больше, больше или равно, меньше, меньше или равно, не	=, >, >=,<,<=, <>

	равно)	
6	Десятичный разделитель	.
7	Комментарий	//

Переменные

Имена переменных во встроенном языке 1С должны обязательно начинаться с буквы русского или латинского алфавита или символа «_», могут содержать цифры и не должны совпадать со служебными конструкциями языка.

Переменная во встроенном языке 1С может быть объявлена явно или неявно.

Для *явного* объявления используется ключевое слово *Перем*, после которого можно указать одну переменную или перечислить несколько. После имени переменной можно указать ключевое слово Экспорт, тогда она будет доступна из других модулей (при соблюдении правил видимости):

```
Перем Цена, Стоимость;
Перем Склад Экспорт;
```

После объявления переменной присваивается значение *Неопределено* – пустое значение, не принадлежащее ни к одному типу данных. Задать значение переменной можно с помощью оператора присваивания:

```
Цена = 100;
```

При *неявном* объявлении имя переменной указывается непосредственно перед использованием:

```
СтавкаНалога = 0.2;
```

Условный оператор

Основным условным оператором во встроенном языке 1С: Предприятие является оператор *Если*, который имеет следующий синтаксис:

```
Если <условие> Тогда
    //операторы
[ИначеЕсли <условие> Тогда
    //операторы]
[Иначе <условие> Тогда
    //операторы]
КонецЕсли;
```

В квадратных скобках приведены необязательные части оператора.

Пример:

```
Если Прибыль < 1000 Тогда
    СтавкаНалога = 0.1
ИначеЕсли Прибыль < 10000 Тогда
    СтавкаНалога = 0.15
Иначе
```

СтавкаНалога = 0.2
КонецЕсли;

Еще какое-либо условие необходимо проверить внутри выражения, то удобно использовать следующую *упрощенную форму условного оператора*:

?(условие, выражение1, выражение2),

где если условие равно Истина, то возвращается выражение1, если же условие равно Ложь, то возвращается выражение2, например:

СтавкаНалога = ?(Прибыль < 1000, 0.1, 0.15);

Для составления более сложных условий можно использовать вложенные операторы:

СтавкаНалога = ?(Прибыль < 1000, 0.1, ?(Прибыль < 10000, 0.15, 0.2));

Циклы

Встроенный язык платформы 1С: Предприятие поддерживает следующие виды циклов:

- цикл со счетчиком *Для*;
- цикл для обхода коллекций значений *Для каждого*;
- цикл с предусловием *Пока*.

Цикл *Для* для контроля числа повторений операторов внутри цикла использует переменную-счетчик и имеет следующий синтаксис:

```
Для ИмяПеременной = Выражение1 По Выражение2 Цикл
    //операторы
[Прервать;]
    //операторы
[Продолжить;]
    //операторы
КонецЦикла;
```

Перед началом выполнения цикла переменной-счетчику ИмяПеременной присваивается значение Выражение1 и далее при каждом повторении цикла ее значение увеличивается на 1. Цикл выполняется до тех пор пока переменная-счетчик будет меньше или равна значению Выражение2. При необходимости изменить значение ИмяПеременной можно внутри цикла.

Оператор *Прервать* позволяет прекратить выполнение цикла в любой его точке и перейти к операторам, следующим после цикла.

Оператор *Продолжить* позволяет пропустить все операторы цикла, расположенные после него, и перейти к вычислению переменной-счетчика и проверке условия выполнения цикла.

Пример:

```
Сумма = 0;
Для Счетчик = 0 По 9 Цикл
    Сумма = Сумма + МассивЭлементов[Счетчик];
КонецЦикла;
```

Цикл *Для каждого* предназначен для последовательного обхода коллекций значений (например, массивов):

```
Для каждого ИмяПеременной Из ИмяКоллекции Цикл
        //операторы
[Прервать;]
        //операторы
[Продолжить;]
        //операторы
КонецЦикла;
```

ИмяПеременной1 – переменная, которой при каждом повторении цикла присваивается очередной элемент коллекции ИмяКоллекции. В теле цикла элемент коллекции доступен как для чтения, так и для изменения.

Пример:

```
Сумма = 0;
Для каждого Элемент Из МассивЭлементов Цикл
        Сумма = Сумма + Элемент;
КонецЦикла;
```

Цикл с предусловием *Пока* имеет следующий синтаксис:

```
Пока <условие> Цикл
        //операторы
[Прервать;]
        //операторы
[Продолжить;]
        //операторы
КонецЦикла;
```

В начале каждого повторения цикла проверяется условие, операторы цикла выполняются до тех пор, пока условие будет равно Истина.

Типы данных встроенного языка

Набор типов, которыми могут оперировать прикладные решения, довольно разнообразен. Он позволяет решать как задачи обработки данных, так и задачи представления этих данных пользователю и интерактивной работы с ними. Можно выделить следующие основные категорий типов данных 1С: Предприятие:

- *примитивные типы* - это такие типы как Строка, Число, Дата, Булево и другие; эти типы не являются чем-то особенным для «1С: Предприятия 8», как правило, такие типы данных существуют и в других программных системах;
- *универсальные коллекции значений* – более сложные типы данных, такие как массив, структура, соответствие, список значений и др.;

- *прикладные типы* – уникальные типы данных, используемые только в данном прикладном решении, например, СправочникСсылка.Номенклатура, ДокументСсылка.ПриходнаяНакладная; экземпляры данных типов наследуют все функциональность прикладных объектов платформы (справочников, документов и др.).

Примитивные типы данных и некоторые функции для работы с ними

Тип *Число* используется для задания любых десятичных чисел, содержащих не более 38 разрядов.

При работе с числами можно использовать стандартные арифметические операции и функции, перечисленные в табл. Необязательные параметры функций заключены в квадратные скобки.

№	Обозначение функции	Назначение
1	Цел(x)	возвращает целую часть числа x
2	Окр(x, [Разрядность], [РежимОкругления])	возвращает результат округления числа x до нужного разряда и в соответствии с указанным режимом округления
3	Sin(x), Cos(x), Tan(x), ASin(y), ACos(y), ATan(y)	стандартные тригонометрические функции: синус, косинус, тангенс, арксинус, арккосинус, арктангенс; угол x задается в радианах; число $y \in [-1; 1]$
4	Log(x), Log10(x)	вычисление логарифма и десятичного логарифма числа x, x > 0
5	Exp(x)	вычисление e^x, где e – основание натурального логарифма
6	Pow(x, a)	возведение числа x в степень a
7	Sqrt(x)	извлечение квадратного корня из числа x, x ≥ 0

Примеры:

```
ВсегоМесяцев = 19;

ЧислоЛет = Цел(ВсегоМесяцев / 12);       // результат: 1
ЧислоМесяцев = ВсегоМесяцев % 12;      // 7

Цена = 12543.655;
ЦенаДоКопеек = Окр(Цена, 2);   // 12543.66
ЦенаДоКопеек = Окр(Цена, 2, РежимОкругления.Окр15как10); // 12543.65

ЦенаДоСотен = Окр(Цена, -2);   // 12500
ЦенаДоТысяч = Окр(Цена, -3);  // 13000
```

Более подробное описание функций можно посмотреть в Синтакс-помощнике, который можно вызвать, выбрав в главном меню конфигуратора команду Справка / Синтакс-помощник или нажав на панели команд конфигуратора на пиктограмму . Фрагмент содержания Синтакс-помощника показан на рис.

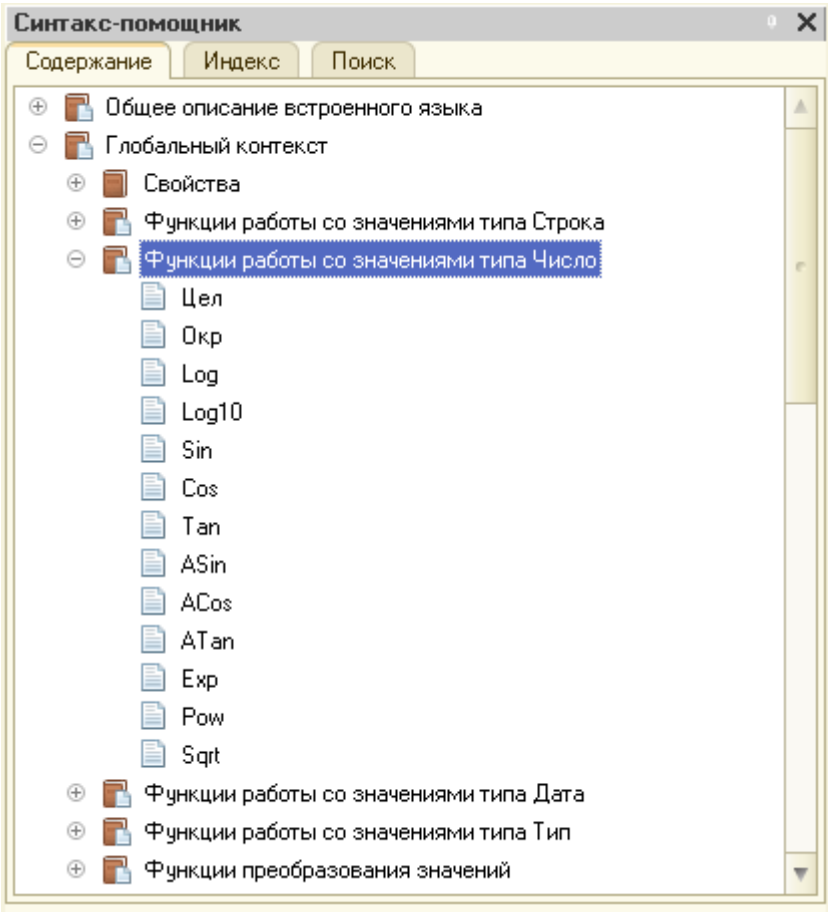


Рис. Фрагмент содержания Синтакс-помощника

Тип *Строка* предназначен для работы с последовательностями символов. Во встроенном языке 1С строка заключается в кавычки:

Стр = "Поле должно быть обязательно заполнено! ";

Если текст строки должен содержать кавычки, то они задаются двойными символами кавычек:

Стр = "ОАО ""Карандаш""";

Переменная типа Строка может содержать строку произвольной длины. Если строка очень длинная, то можно записать ее в несколько строк с помощью одного из следующих способов:

- каждую часть строки записать в кавычках, при этом между подстроками не должно быть никаких символов, кроме пробелов, переводов строк и комментариев:

ТекстЗапроса = "ВЫБРАТЬ Номенклатура.Наименование "
"ИЗ Справочник.Номенклатура КАК Номенклатура";

- в начале каждой подстроки помещать символ перевода строки «|»:

ТекстЗапроса = "ВЫБРАТЬ Номенклатура.Наименование

| ИЗ Справочник.Номенклатура КАК Номенклатура";

Основные операции и функции обработки строк перечислены в табл.

№	Операция или функция	Назначение
1	+	конкатенация (объединение) строк
2	СтрДлина(Стр)	возвращает количество символов в строке
3	СокрЛ(Стр), СокрП(Стр), СокрЛП(Стр)	возвращает строку, полученную путем удаления незначащих символов слева, справа, слева и справа от первого значащего символа строки Стр
4	Лев(Стр, ЧислоСимволов), Прав(Стр, ЧислоСимволов), Сред(Стр, НачальныйНомер, [ЧислоСимволов])	возвращает подстроку длиной ЧислоСимволов, полученную путем копирования из строки Стр, начиная с первого символа слева, справа или с символа с номером НачальныйНомер
5	СтрНайти(Стр, ПодстрокаПоиска, [НаправлениеПоиска], [НачальнаяПозиция], [НомерВхождения])	возвращает позицию первого символа подстроки ПодстрокаПоиска в строке Стр; дополнительно можно задать параметры НаправлениеПоиска, НачальнаяПозиция, НомерВхождения
6	ВРег(Стр), НРег(Стр), ТРег(Стр)	возвращает подстроку, полученную путем приведения символов исходной строки к верхнему, нижнему или титульного регистру (первая буква каждого слова строки заглавная, остальные – строчные)
7	Символ(КодСимвола), КодСимвола(Стр, [НомерСимвола])	функции получения символа по его коду КодСимвола в кодировке Unicode и кода символа строки с номером НомерСимвола (нумерация символов начинается с 1)
8	ПустаяСтрока(Стр)	возвращает значение Истина, если строка Стр не содержит значащих символов или не содержит символов вообще, и Ложь в противном случае
9	СтрЗаменить(Стр, ПодстрокаПоиска, ПодстрокаЗамены)	возвращает строку, полученную из строки Стр путем замены подстроки ПодстрокаПоиска на строку ПодстрокаЗамены
10	СтрСравнить(Стр1, Стр2)	выполняет сравнение строк без учета регистра; возвращает -1, если Стр1 < Стр2, 1, если Стр1 > Стр2, 0, если Стр1 = Стр2
11	СтрНачинаетсяС(Стр, СтрокаПоиска), СтрЗаканчиваетсяНа(Стр, СтрокаПоиска)	возвращает значение Истина, если строка Стр начинается с предполагаемой строки СтрокаПоиска (или заканчивается на нее), и Ложь в противном случае
12	СтрРазделить(Стр, Разделитель, [ВключатьПустые]), СтрСоединить(Стр, Разделитель)	функции разделения и соединения строк; первая функция возвращает массив строк, которые получились в результате разделения строки Стр на подстроки, отделенные друг от друга строкой Разделитель; вторая функция соединяет массив строк в строку

		Стр, отделяя подстроки заданной строкой Разделитель

Тип *Дата* предназначен для работы с переменными, содержащими дату и время. С помощью переменной данного типа можно задать любую дату в промежутке с 01.01.0001 г. по 31.12.3999 г.

Чтобы присвоить переменной типа Дата конкретное значение, нужно задать строку в одинарных кавычках в формате 'ГГГГММДДччммсс', где ГГГГ – четыре цифры года, ММ – две цифры месяца, ДД – две цифры дня, чч – часы, мм – минуты, сс – секунды; время указывать не обязательно, в этом случае оно будет равно 00:00:00; для удобства ввода можно использовать любые разделители:

```
ДатаСоздания = '20200223123725';          // 23.02.2020 12:37:25
ДатаПроверки = '20200224';                // 24.02.2020 00:00:00
ДатаПриема = '202005121615';              // 12.05.2020 16:15:00
ДатаЗаказа = '2018-06-03 12:37:25';       // 03.06.2018 12:37:25
ДатаПодтверждения = '2018/09/13 16-17-05'; // 13.09.2018 16:17:05
ДатаДоставки = '2020 08 14';              // 14.08.2020 00:00:00
ДатаНачала = '0001 01 01';          // 01.01.0001 00:00:00 (пустая дата)
```

Операции и основные функции для работы с датами приведены в табл.

№	Операция или функция	Назначение
1	+, -	определены в формате Дата + Число, Дата – Число; прибавляют к дате (отнимают от нее) число секунд
2	Дата(Стр), Дата(Год, Месяц, День, [Час], [Минута], [Секунда])	возвращают дату, полученную путем преобразования строки с датой Стр или путем соединения чисел - отдельных компонент даты
3	ТекущаяДата()	возвращает текущую системную дату на компьютере
4	Год(Дата), Месяц(Дата), День(Дата), Час(Дата), Минута(Дата), Секунда(Дата)	возвращают число - отдельную указанную компоненту даты Дата
5	НачалоГода(Дата), НачалоКвартала(Дата), НачалоМесяца(Дата), НачалоНедели(Дата), НачалоДня(Дата), НачалоЧаса(Дата), НачалоМинуты(Дата), НачалоСекунды(Дата)	возвращают дату начала указанного периода для даты Дата
6	КонецГода(Дата), КонецКвартала(Дата), КонецМесяца(Дата), КонецНедели(Дата), КонецДня(Дата), КонецЧаса(Дата), КонецМинуты(Дата), КонецСекунды(Дата)	возвращают дату конца указанного периода для даты Дата
7	НеделяГода(Дата), ДеньГода(Дата)	возвращают для даты Дата число - номер недели (дня) в году

8	ДеньНедели(Дата)	возвращает число – номер дня в неделе для даты Дата
9	ДобавитьМесяц(Дата, ЧислоМесяцев)	возвращает дату, полученную после добавления к дате Дата числа ЧислоМесяцев

Примеры:

```
ДатаСоздания = ТекущаяДата();
ДатаПроверки = Дата("20200512161505");        //12.05.2020 16:15:05
ДатаПриема = Дата(2020, 10, 16);              //16.10.2020 00:00:00
ПустаяДата = Дата(1, 1, 1);                   //01.01.0001 00:00:00
ТекущееЧисло = День(ТекущаяДата());
ТекущаяНеделя = НеделяГода(ТекущаяДата());
ДатаЧерезЧас = ТекущаяДата() + 60 * 60;
```

Переменные типа *Булево* могут принимать только два значения: Истина и Ложь. Значения данного типа возвращаются в качестве выполнения, например, операций сравнения, и могут быть использованы в условиях, циклах и т.д.

Функции преобразования примитивных типов данных

Во встроенном языке 1С для преобразования типов данных переменных могут использовать два способа:

- *неявное преобразование* – осуществляется самой системой при выполнении арифметических операций над переменными различных типов, при этом система пытается привести все переменные в правой части выражения к типу первой из них;
- *явное преобразование* – разработчик может явно указать, к какому типу нужно преобразовать переменную; для этого используются функции, одноименные с названиями типов данных, Число(), Строка(), Дата(), Булево().

Примеры:

```
Количество = 5 + "14";                        //19
Склад = "Склад №" + 3;                        //"Склад№3"
СтрТекДата = "Дата: " + ТекущаяДата();        // Дата: 23.08.2020  08:00:33
Налог = 5 + 0.3 * Истина;                     //5.3
СтрУчетПоСкладам = "УчетПоСкладам: " + Истина;    //УчетПоСкладам: Да
Цена = Число("12") + "0.5";                   //12.5
СтрНомерПКО = Строка(00000123) + "04П";    //"0000012304П"
```

Универсальные коллекции значений

Универсальные коллекции значений представляют собой более сложные типы данных: массив, структура, соответствие, список значений, таблица значений, дерево значений, фиксированный массив.

Коллекции значений являются динамическими структурами, создаваемыми разработчиками для выполнения каких-либо алгоритмов обработки данных. В отличие от объектов конфигурации коллекции существуют только во время исполнения процедуры, в которой они описаны и в базе данных не сохраняются.

Все универсальные коллекции, независимо от вида, обладают рядом общих свойств:

- новые экземпляры коллекции создаются с помощью конструктора *Новый*, параметры которого отличаются в зависимости от вида коллекции;
- к элементу коллекции можно обратиться по его номеру, при этом нумерация начинается с 0;
- обход элементов коллекции возможен с помощью циклов Для и Для каждого из;
- для большинства универсальных коллекций применимы такие методы, как *Количество(), Индекс(), Добавить(), Вставить(), Удалить()* и *Найти()*.

Далее подробнее рассмотрим виды универсальных коллекций.

Массив – пронумерованная коллекция значений произвольного типа, к элементам которого можно обращаться по индексу, указываемому в квадратных скобках; элементом массива может быть, в том числе, и другой массив.

Для обхода массива можно использовать цикл *Для каждого* или циклы *Для*, *Пока* с перебором индексов элементов.

Конструкторы и основные методы перечислены в табл.

№	Метод	Описание
1	Новый Массив([КоличествоЭлементов1] [КоличествоЭлементов2…), Новый Массив([ФМ])	методы-конструкторы – создают новый массив на основании числа элементов КоличествоЭлементов1 или фиксированного массива ФМ; если число элементов не задано, создается одномерный массив без элементов, если указано несколько параметров, то создается многомерный массив
2	ВГраница()	возвращает наибольший индекс в массиве
3	Количество()	возвращает количество элементов в массиве
4	Вставить(Индекс, [Значение]), Добавить([Значение])	вставляет элемент Значение или, если он не задан, то Неопределено на позицию с номером Индекс (метод Вставить()) или в конец массива (метод Добавить())
5	Найти(Значение)	возвращает индекс элемента Значение; если элемент не найден, то возвращает Неопределено
6	Очистить()	удаляет все элементы из массива
7	Получить(Индекс)	возвращает элемент с номером Индекс, аналогичен оператору [Индекс]
8	Удалить(Индекс)	удаляет элемент с индексом Индекс
9	Установить(Индекс, Значение)	устанавливает значение по индексу Индекс

Примеры:

```
НовыйМассив = Новый Массив(3);                    //{ , , }

Для Индекс = 0 По НовыйМассив.ВГраница() Цикл
        НовыйМассив[Индекс] = 1;
КонецЦикла;                                       //{1, 1, 1}

НовыйМассив.Добавить(Истина);                     //{1, 1, 1, Истина}

Для Индекс = 0 По НовыйМассив.Количество() - 1 Цикл
        НовыйМассив[Индекс] = "12";
КонецЦикла;                                       //{"12", "12", "12", "12"}
```

Фиксированный массив, в отличии от обычного массива, заполняется один раз при создании и далее не может быть изменен. Конструктор и основные методы приведены в табл.

№	Метод	Описание
1	Новый ФиксированныйМассив(ФМ)	метод-конструктор – создает новый фиксированный массив из обычного массива ФМ
2	ВГраница(), Количество(), Найти(Значение), Получить(Индекс)	аналогичны одноименным функциям для обычного массива

Структура – коллекция элементов, имеющих специальный тип *КлючИЗначение* и включающих два поля:

- ключ – строковый идентификатор элемента;
- значение – данные произвольного типа, соответствующие ключу.

Таким образом, каждое отдельное значение структуры имеет не номер, как в массиве, а имя (ключ).

Для доступа к значениям структуры указывается имя переменной (экземпляра структуры), и через точку имя ключа:

ИмяСтруктуры.ИмяКлюча

Обход коллекции возможен с помощью цикла *Для каждого*.

Методы структуры перечислены в табл.

№	Метод	Описание
1	Новый Структура(ФС), Новый Структура(СтрСписокКлючей, [СписокЗначений])	методы-конструкторы – создают новую структуру на основании описания элементов фиксированной структуры ФС или строки СтрСписокКлючей и списка СписокЗначений; если СписокЗначений не задан, то значения

		всех ключей равны Неопределено
2	Свойство(ИмяКлюча, [НайденноеЗначение])	возвращает Истина, если в структуре существует ключ ИмяКлюча (или пара ИмяКлюча - НайденноеЗначение), Ложь – в противном случае
3	Удалить(ИмяКлюча)	удаляет элемент с ключом ИмяКлюча
4	Вставить(ИмяКлюча, [Значение]), Количество(), Очистить()	аналогичны методам для других коллекций значений

Примеры:

```
ДанныеОбОрганизации = Новый Структура(
        "Наименование, ИНН, ДатаРегистрации, УставныйКапитал",
        "ОАО Карандаш", "28466290", '20161127', 10000);

Налоги = Новый Структура(
        "НДС, НалогНаПрибыль, НалогНаИмущество");

Налоги.НДС = 10;
Налоги.НалогНаПрибыль = 20;
Налоги.НалогНаИмущество = 2;
```

Фиксированная структура – структура, недоступная для изменения. Набор методов аналогичен обычной структуре, за исключением функций редактирования.

Соответствие - коллекция элементов типа *КлючИЗначение*, в которых ключ может быть произвольного типа. Для обхода коллекции используется цикл Для каждого … из … Цикл; возможно использование оператора [] с указанием значения ключа элемента.

Методы соответствия перечислены в табл.

№	Метод	Описание
1	Новый Соответствие(), Новый Соответствие([ФС])	методы-конструкторы – создают новое пустое соответствие или с элементами из исходного фиксированного соответствия ФС
2	Вставить(ИмяКлюча, [Значение]), Количество(), Очистить(), Получить(ИмяКлюча), Удалить(ИмяКлюча)	аналогичны методам для других коллекций значений

Пример:

```
КодыРегионов = Новый Соответствие();

КодыРегионов.Вставить(36, "Воронежская область");
КодыРегионов.Вставить(37, "Ивановская область");
КодыРегионов.Вставить(38, "Иркутская область");
```

Фиксированное соответствие – неизменяемая коллекция пар КлючИЗначение, где ключ может быть любого типа.

Список значений – коллекция элементов типа *ЭлементСпискаЗначений*. Каждый элемент списка имеет четыре свойства:

- Значение – содержит данные произвольного типа;
- Картинка – содержит картинку, связанную с данным элементом списка;
- Пометка – логическое значение (Истина или Ложь), связанное с элементом списка;
- Представление – внешнее представление элемента списка (строка).

Для обращения к элементу списка возможно использование оператора []. Обход коллекции осуществляется цикл *Для каждого*.

Основные методы списка значений перечислены в табл.

№	Метод	Описание
1	Новый СписокЗначений()	метод-конструктор – создает новый список значений
2	ВыгрузитьЗначения(), ЗагрузитьЗначения(МассивЗначений)	функции выгрузки / загрузки значений в массив, с помощью которых можно получить массив со значениями элементов списка или загрузить значения из массива МассивЗначений
3	ЗаполнитьПометки(ЗначениеПометки)	устанавливает пометку в значение ЗначениеПометки у всех элементов списка
4	Сдвинуть(Элемент, Смещение)	сдвигает Элемент на указанное число позиций вперед, если Смещение больше нуля, или назад, если Смещение меньше нуля; параметр Элемент – индекс элемента либо сам элемент
5	Скопировать()	возвращает список значений – копию заданного списка
6	СортироватьПоЗначению([Направление]), СортироватьПоПредставлению([Направление])	функции сортировки списков по свойства Значение или Представление; Направление – способ сортировки, по умолчанию сортировка по возрастанию
7	Вставить(Индекс, [Значение], [Представление], [Пометка], [Картинка]), Добавить([Значение],	аналогичны методам для других коллекций значений

	[Представление], [Пометка], [Картинка]), Количество(), Очистить(), Получить(Индекс), Удалить(Элемент)	

Пример:

```
СписокТовары = Новый СписокЗначений();

СписокТовары.Добавить("006523", "Карандаши", Ложь);
СписокТовары.Добавить("006552", "Ручки", Ложь);
СписокТовары.Добавить("004283", "Тетради", Истина);
СписокТовары.СортироватьПоПредставлению(НаправлениеСортировки.Убыв).
```

Таблица значений предназначена для хранения произвольных данных в табличном виде и представляет собой коллекцию элементов типа *СтрокаТаблицыЗначений*.

Таблица значений имеет два свойства:

- свойство *Индексы* содержит коллекцию индексов таблицы значений;
- свойство *Колонки* возвращает коллекцию типа *КоллекцияКолонокТаблицыЗначений*, для которой доступны такие стандартные методы, как *Вставить(), Добавить(), Индекс(), Количество(), Найти(), Очистить(), Сдвинуть(), Удалить()* и др.

Отдельная колонка таблицы представляет собой элемент типа *КолонкаТаблицыЗначений* и имеет такие свойства, как *Заголовок(), Имя(), ТипЗначения(), Ширина(),* с помощью которых можно установить и получить соответствующие параметры колонок.

Чтобы обратиться к отдельному элементу коллекции строк или столбцов таблицы значений, можно использовать оператор [], в котором указывается индекс колонки или строки, или цикл *Для каждого*. К отдельной колонке в коллекции колонок или в строке таблицы также возможно обращение напрямую по имени (через точку).

Нумерация индексов строк и колонок в таблице начинается с 0.

Основные методы таблицы значений перечислены в табл.

№	Метод	Описание
1	Новый ТаблицаЗначений()	метод-конструктор – создает новую таблицу значений
2	ВыгрузитьКолонку(Колонка), ЗагрузитьКолонку(ИмяМассива, Колонка)	функции выгрузки данных из колонки таблицы в массив и загрузки данных из массива; в качестве параметра Колонка может быть указана сама колонка, ее индекс или имя
3	Итог(ИмяКолонки)	возвращает итог для колонки с именем ИмяКолонки
4	Сортировать(СтрКолонки,	сортирует значения в колонках

	[ОбъектСравнения])	таблицы, перечисленных через запятую в строке СтрКолонки, после имени столбца в строке можно указать направление (Возр – по умолчанию или Убыв); элементы, чьи типы не совпадают, сравниваются по коду типа, а элементы простых типов сравниваются по значению
5	Вставить(Индекс), Найти(Значение, [Колонки]), Очистить(), Получить(ИндексСтроки), Скопировать([СтрСтроки], [СтрКолонки]), Удалить(Строка)	аналогичны методам для других коллекций значений

Пример:

```
Таб = Новый ТаблицаЗначений;

Таб.Колонки.Добавить("Товар");
Таб.Колонки.Добавить("Цена");

НоваяСтрока = Таб.Добавить();
НоваяСтрока.Товар = "Карандаш";
НоваяСтрока.Цена = 5.4;

Для Каждого СтрокаТаблицы Из Таб Цикл
        СтрокаТаблицы.Цена = СтрокаТаблицы.Цена * 1.05;
КонецЦикла;
```

Дерево значений представляет собой таблицу значений произвольного типа, в которой строки могут образовывать иерархические структуры.

Дерево значений имеет два свойства:

- *Колонки* – содержит коллекцию типа *КоллекцияКолонокДереваЗначений*, содержащую колонки - элементы типа *КолонкаДереваЗначений*;

- *Строки* – коллекция типа *КоллекцияСтрокДереваЗначений* – набор строк первого уровня иерархии.

Каждая строка первого уровня, в свою очередь, является коллекцией элементов типа *СтрокаДереваЗначений*, которые обладают следующими свойствами и методами:

- свойство *Родитель* определяет строку, которой подчинена данная строка таблицы; для корневого элемента дерева свойство Родитель имеет значение Неопределено;

- методы *Вставить()*, *ВыгрузитьКолонку()*, *Добавить()*, *ЗагрузитьКолонку()*, *Итог()*, *Количество()*, *Найти()*, *Очистить()*, *Сортировать()*, *Удалить()* и др.

Дерево значений имеет метод *Скопировать()*, позволяющий создать полную копию исходного дерева.

Пример:

```
ДеревоЗнч = Новый ДеревоЗначений;

ДеревоЗнч.Колонки.Добавить("Наименование");

СтрокаУровень1 = ДеревоЗнч.Строки.Добавить();
СтрокаУровень1.Наименование = "Тетради";

СтрокаУровень2 = СтрокаУровень1.Строки.Добавить();
СтрокаУровень2.Наименование = "Тетрадь 12 л.";
```

Функции для определения и сравнения типов

Для идентификации типов данных переменных могут использоваться следующие функции:

- Тип(ИмяТипа) – возвращает тип данных для строки ИмяТипа;
- ТипЗнч(Значение) – возвращает тип переменной Значение.

Пример условия на сравнение типов:

```
ТипЗнч(Склад) = Тип("Строка").
```

Функции для интерактивной работы

№	Метод	Описание
1	Сообщить(ТекстСообщения, [Статус])	выводит сообщение, задаваемое строкой ТекстСообщения, в окно сообщений
2		

Процедуры и функции

Процедуры и функции на встроенном языке платформы имеют следующий синтаксис:

```
Процедура ИмяПроцедуры([[Знач] Параметр1 [= Значение1], …]) [Экспорт]
    //объявления локальных переменных
    //операторы
    [Возврат;]
    //операторы
КонецПроцедуры;
Функция ИмяФункции([[Знач] Параметр1 [= Значение1], …]) [Экспорт]
    //объявления локальных переменных
    //операторы
    Возврат ВозвращаемоеЗначение;
    //операторы
```

КонецПроцедуры;

Процедуры и функции могут быть описаны как без параметров, так и с произвольным количеством параметров. По умолчанию параметры в подпрограмму передаются *по ссылке*, т.е. изменение параметра внутри процедуры или функции приводит к изменению передаваемого фактического параметра. Если такое изменение не нужно, то перед именем параметра необходимо указать необязательное ключевое слово *Знач*. В этом случае фактический параметр будет передан *по значению* и не изменится при любых манипуляциях с ним внутри подпрограммы.

При необходимости любому параметру процедуры или функции, независимо от его расположения в списке, может быть присвоено значение по умолчанию:

Параметр1 = Значение1.

Необязательное ключевое слово *Экспорт* записывается после заголовка подпрограммы и указывает на то, что данная процедура (функция) может быть вызвана из других программных модулей (при соблюдении правил видимости).

Функция, в отличие от процедуры, может возвращать значение. Для этого используется ключевое слово *Возврат*, после которого указывается само значение. Если функция не содержит данную конструкцию, то она возвращает значение *Неопределено*. Слово *Возврат* может быть использовано и внутри процедуры, однако здесь оно указывается не для возврата какого-либо значения, а для принудительного выхода из процедуры и перехода к оператору, следующему за вызовом подпрограммы.

Директивы компиляции программного кода

Программный код на языке 1С перед исполнением компилируется, т.е. преобразуется в специальный код. В случае клиент-серверного приложения компиляция каждого модуля производится отдельно на клиенте и на сервере. Один и тот же модуль управляемого приложения может содержать участки кода, которые возможно выполнить только на клиенте или на сервере. Например, на сервере невозможно обработать события формы, а на клиенте нельзя выполнить запрос к базе данных. Для указания места исполнения процедуры или функции используются следующие специальные инструкции - *директивы компиляции*:

- *&НаКлиенте* - указывает на то, что подпрограмма выполняется на стороне клиента, где будут доступны вызовы *любых* процедур модуля формы, а также элементы, параметры и реквизиты самой формы;
- *&НаСервере* – процедура (функция) будет выполнена на сервере, доступны данные формы, есть возможность обращаться к базе данных, нельзя обратиться к процедурам на клиенте;
- *&НаСервереБезКонтекста* - процедура (функция) выполняется на сервере и не имеет доступа к реквизитам формы.

Особенности программной работы с данными информационной базы

Разработчик прикладных решений на платформе 1С: Предприятие хотя и не имеет прямого доступа к базе данных приложения, но может выполнять все необходимые операции с данными информационной базы с помощью встроенных механизмов платформы:

- описывать структуры данных в конфигураторе;
- манипулировать данными с помощью объектов встроенного языка;
- составлять запросы к данным.

Для доступа к данным информационной базы с помощью встроенного языка используются методы так называемого *глобального контекста* - специального набора свойств, процедур и функций, доступных в любом программном модуле конфигурации.

Свойства глобального контекста (*Документы*, *Константы*, *Справочники* и т.д.) позволяют получить доступ к объектам, определенным в конфигурации. Процедуры и функции глобального контекста предназначены для работы с переменными различных типов данных, формами, информационной базой, внешними компонентами, файлами, операционной системой и т.д.

Модель базы данных платформы 1С: Предприятие, с которой работает разработчик, поддерживает два способа доступа к данным:

- *объектный* – обращение к объекту конфигурации выполняется с помощью объектов встроенного языка, данный способ позволяет выполнять с объектами базы операции чтения и записи;
- *табличный* – объекты конфигурации представляются как набор таблиц, к которым можно обращаться с помощью запросов, этот способ предназначен только для чтения объектов из информационный базы.

Объектная модель доступа к объектам информационной базы

Объектная модель для доступа к объектам информационной базы использует *менеджеры* – специальные программные объекты, содержащие набор свойств и методов для манипулирования объектами конфигурации.

В конфигурации определены менеджеры следующих типов:

- менеджеры коллекций объектов – объекты типов *СправочникиМенеджер*, *ДокументыМенеджер* и др., предназначенные для работы с коллекциями объектов; доступ к ним можно получить из свойств глобального контекста *Справочники*, *Документы* и т.д.;
- менеджеры объектов – менеджеры типов *СправочникМенеджер*, *ДокументМенеджер* и др., являющиеся элементами соответствующих коллекций; доступ к отдельному объекту коллекции можно получить напрямую по имени объекта, указав его через точку или внутри оператора [], а также в цикле *Для каждого*.

Примеры обращения к справочникам:

```
Справочники.Номенклатура
Справочники["Номенклатура"]

Для каждого Справочник Из Справочники Цикл
```

…
КонецЦикла;

Менеджеры объектов содержат свойства и методы, предназначенных для управления соответствующим объектом конфигурации. Точный набор свойств и методов и состав их параметров зависит от конкретного объекта.

Например, *СправочникМенеджер* содержит методы *Выбрать()*, *ВыбратьИерархически()*, *НайтиПоНаименованию()*, *НайтиПоРеквизиту()*, *СоздатьЭлемент()* и др., *ДокументМенеджер* включает такие методы, как *Выбрать()*, *НайтиПоНомеру()*, *СоздатьДокумент()* и т.д.

Основными недостатками объектной модели являются ограничение выборки по условиям (нельзя задать сложные условия) и невозможность сделать выборку сразу к нескольким объектам.

Табличная модель доступа к данным

Табличная модель для чтения данных из информационной базы использует специальный объект *Запрос*. Основные свойства и методы данного объекта перечислены в табл.

№	Метод	Описание
1	Новый Запрос([ТекстЗапроса])	метод-конструктор – создает новый запрос, для которого можно сразу указать ТекстЗапроса
2	Текст(ТекстЗапроса)	устанавливает для запроса текст, заданный в строке ТекстЗапроса
3	Выполнить()	выполняет запрос к базе данных
4	УстановитьПараметр(ИмяПараметра, ЗначениеПараметра)	устанавливает параметр запроса, т.е. присваивает значение переменной, используемой в условии запроса

Пример:

Запрос возвращает результат типа *РезультатЗапроса*, основные свойства и методы которого перечислены в табл.

№	Свойство или метод	Описание
1	Колонки()	возвращает коллекцию колонок результата запроса
2	Выбрать([ТипОбхода], [Группировки], [ГруппировкиДляЗначенийГруппировок])	формирует выборку записей из результата запроса
3	Выгрузить([ТипОбхода])	выгружает результат запроса в таблицу значений (по умолчанию) или в дерево значений

Работа с выборкой

Выборку можно получить с помощью метода Выбрать() из данных объекта или результата запроса. Основные свойства и методы выборки перечислены в табл., точный набор свойств и методов зависит от конкретного объекта, для которого она выполнена.

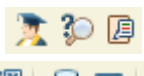
№	Свойство или метод	Описание
1	<ИмяРеквизита>, <ИмяТабличнойЧасти>	свойства позволяют обратиться напрямую к реквизиту или табличной части по их имени (через точку)
2	Код(), Наименование()	возвращают код и наименование элемента объекта
3	ПолучитьОбъект()	получает объект для чтения, изменения, добавления или удаления элемента из текущей позиции выборки; после изменения объекта обязательно надо вызвать метод объекта Записать()
4	Следующий()	получает следующий элемент выборки и возвращает значение Истина, если элемент выбран, или Ложь, если достигнут конец выборки

Пример:

```
НоменклатураВыборка = Справочники.Номенклатура.Выбрать();

Пока НоменклатураВыборка.Следующий() Цикл

    НоменклатураОбъект = НоменклатураВыборка.ПолучитьОбъект();
    НоменклатураОбъект.ВидНоменклатуры = "Товар";
    НоменклатураОбъект.Записать();

КонецЦикла;
```

Возможности конфигуратора для работы с программным кодом

Если в конфигураторе открыть текст какого-либо программного модуля, то на панели команд кроме панели Конфигурация, появятся две дополнительные панели – Модуль и Точки останова (рис.).

а) 

б) 

в) 

г) 

Рис. Панели инструментов конфигуратора: а – Стандартная (фрагмент),

б - Конфигурация, в – Модуль, г – Точки останова

На панели Стандартная расположены следующие команды, полезные для написания модулей на встроенном языке:

- Синтакс-помощник – команда вызова окна помощника, содержащего необходимую информацию по синтаксису и объектам языка программирования платформы 1С (рис);
- Поиск строки в Синтакс-помощнике;
- Открыть окно шаблонов текста (рис.) – шаблоны, содержащиеся в данном окне, можно скопировать и вставить в нужное место программного кода.

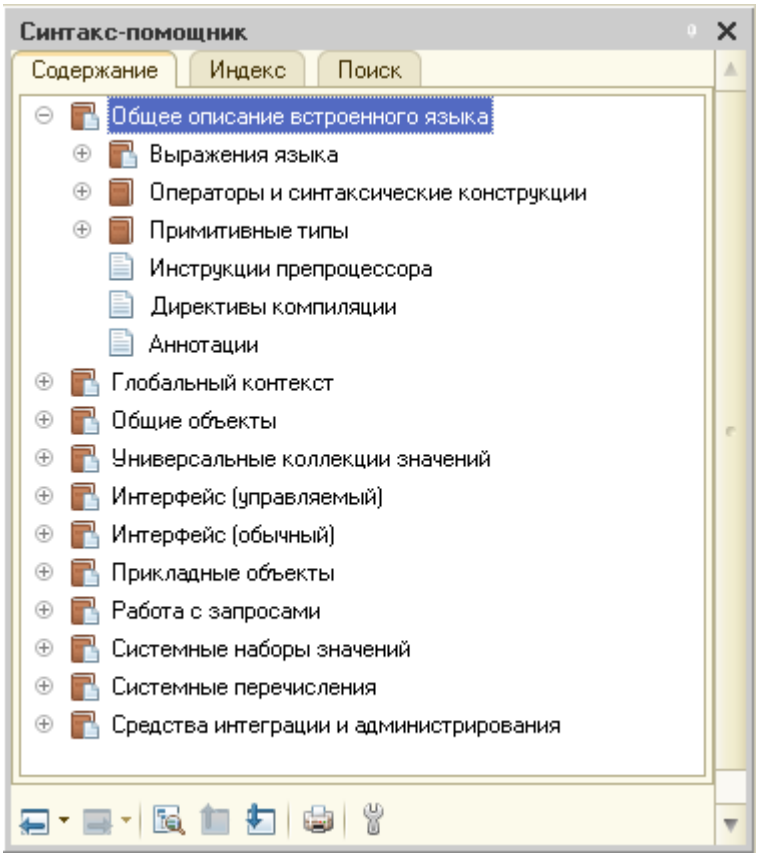


Рис. Содержание Синтакс-помощника

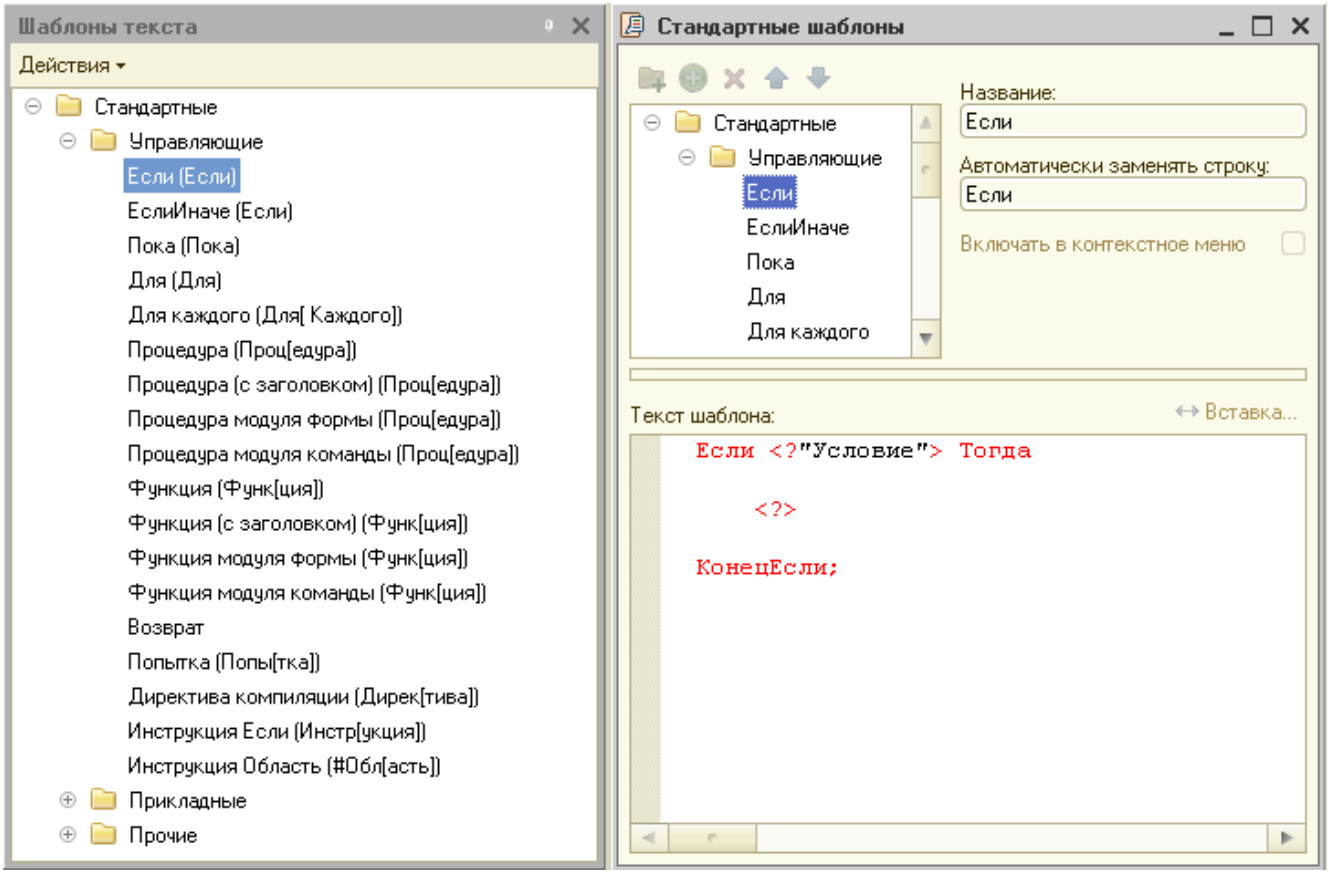


Рис. Диалоговые окна с шаблонами конструкций языка программирования

Панель Конфигурация содержит следующие команды:
- Открыть конфигурацию;
- Закрыть конфигурацию;
- Обновить конфигурацию базы данных;
- Окно конфигурации;
- Начать отладку.

Панель Модуль включает следующие команды:
- Свернуть все группы;
- Развернуть все группы;
- Добавить комментарий;
- Удалить комментарий;
- Проверка модуля;
- Процедуры и функции.

На панели Точки останова расположены следующие команды:
- Точка останова;
- Точка останова с условием;
- Включить/отключить точку останова;
- Удалить все точки останова.

Обработки и команды

При реализации алгоритмов обработки данных на встроенном языке программирования очень часто используются такие объекты конфигурации, как обработки и команды.

Обработки – это прикладные объекты конфигурации, предназначенные для реализации различных вспомогательных алгоритмов обработки данных.

Команда – объект конфигурации, позволяющий создать такой визуальный элементы формы, как кнопка или пункт меню, и связать его с процедурой-обработчиком.

Программное изменение данных справочника с помощью объектной модели

Создадим перечисление ВидНоменклатуры со значениями Товар и Услуга и свяжем его с подсистемой Предприятие (рис.).

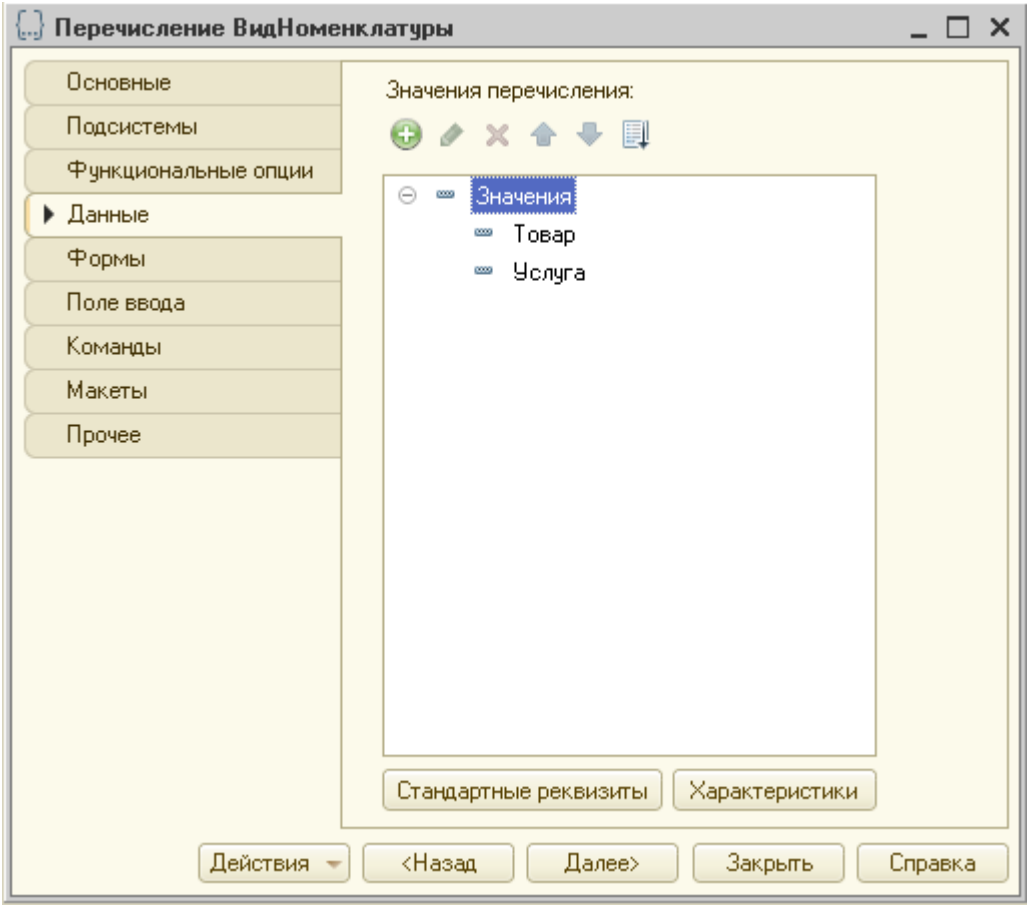


Рис. Данные перечисления ВидНоменклатуры

В справочник Номенклатура добавим реквизит ВидНоменклатуры типа ПеречислениеСсылка.ВидНоменклатуры.

Далее необходимо заполнить новое поле у всей номенклатуры. Но т.к. наименований в справочнике может быть достаточно много, то сделаем это программно.

Создадим в дереве конфигурации новую обработку Служебные обработки и добавим ее в подсистему Предприятие.

На вкладке Формы добавим форму обработки. В редакторе форм в области с реквизитами формы перейдем на вкладку Команды и создадим команду УстановитьВидНоменклатуры (рис.).

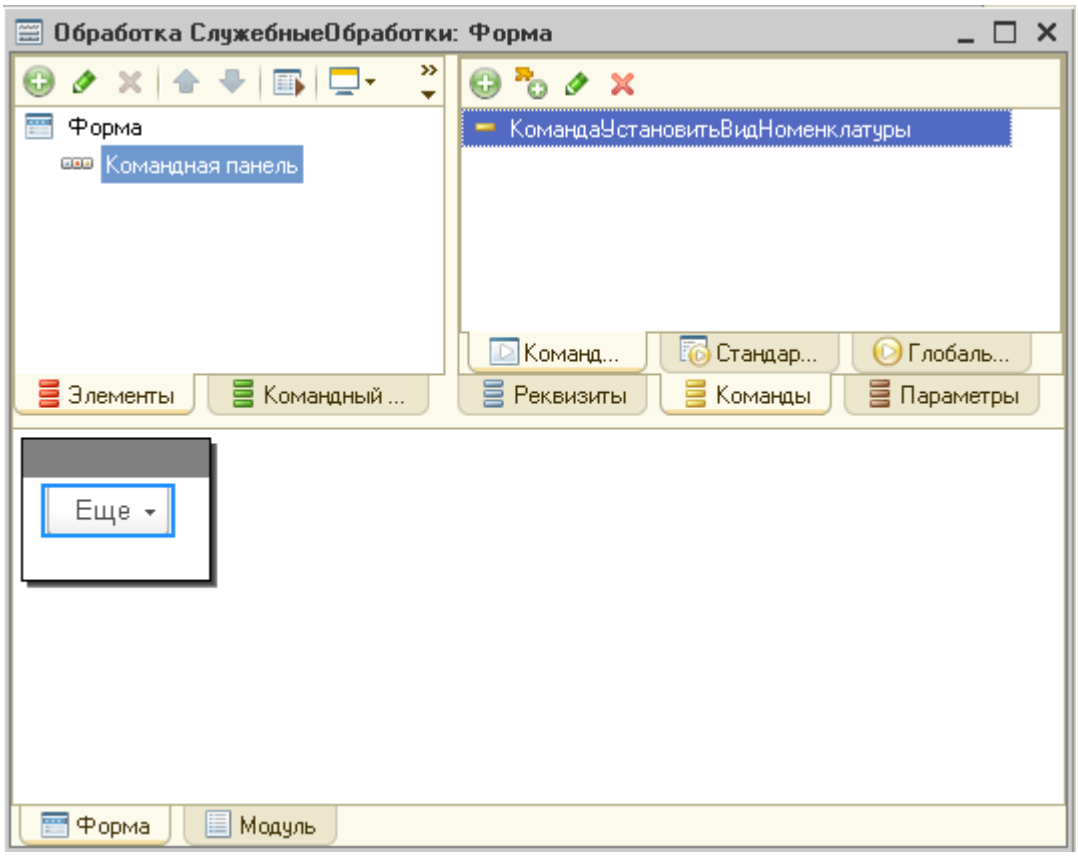


Рис. Редактор формы обработки после добавления команды

В панели свойств команды в поле Действие нажмем на пиктограмму Открыть. В окне выбора типа обработчика действия выберем пункт Создать на клиенте и процедуру на сервере без контекста (рис.). Такой вариант выбран потому, что необходимо будет сделать выборку из справочника, а его данные хранятся в базе данных на сервере, при этом никаких данные с формы считывать не надо будет.

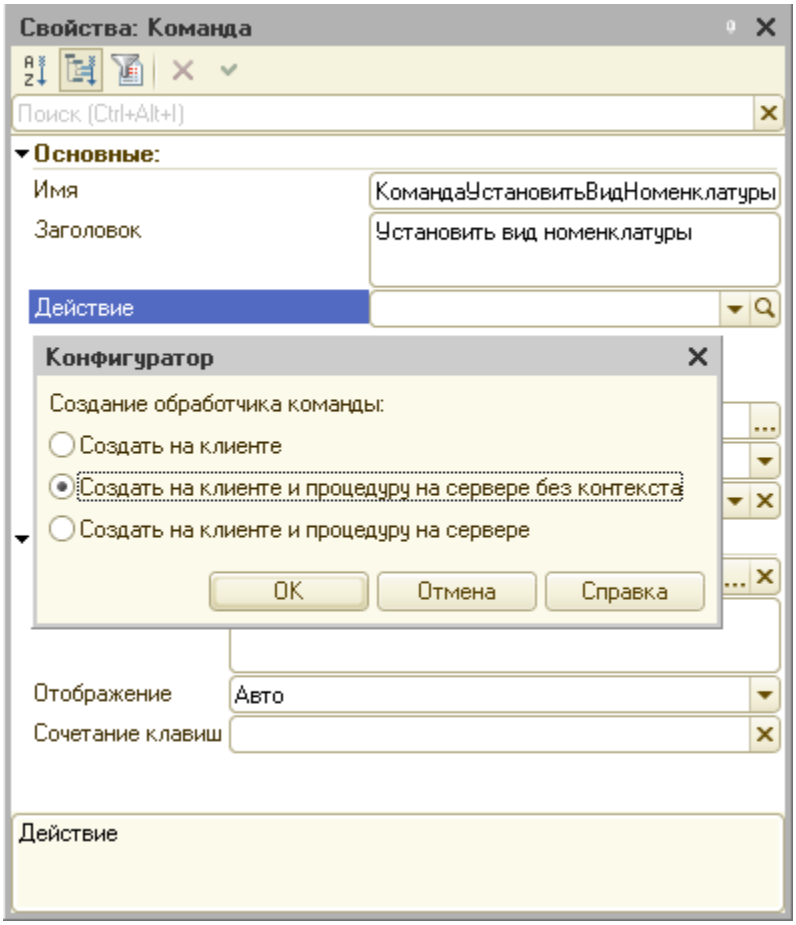


Рис. Выбор места исполнения программного кода команды

При нажатии на кнопку ОК автоматически откроется модуль формы с двумя процедурами, одна из которых исполнится на клиенте, а вторая – не сервере, в клиентской процедуре происходит вызов процедуры с сервера (рис.).

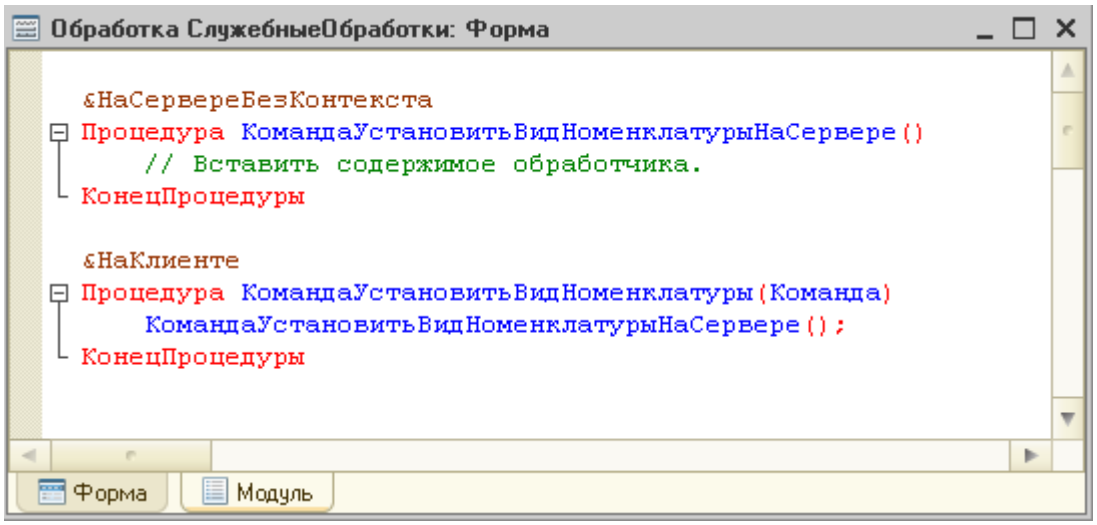


Рис. Процедуры обработки команды

В серверной процедуре напишем следующий код:

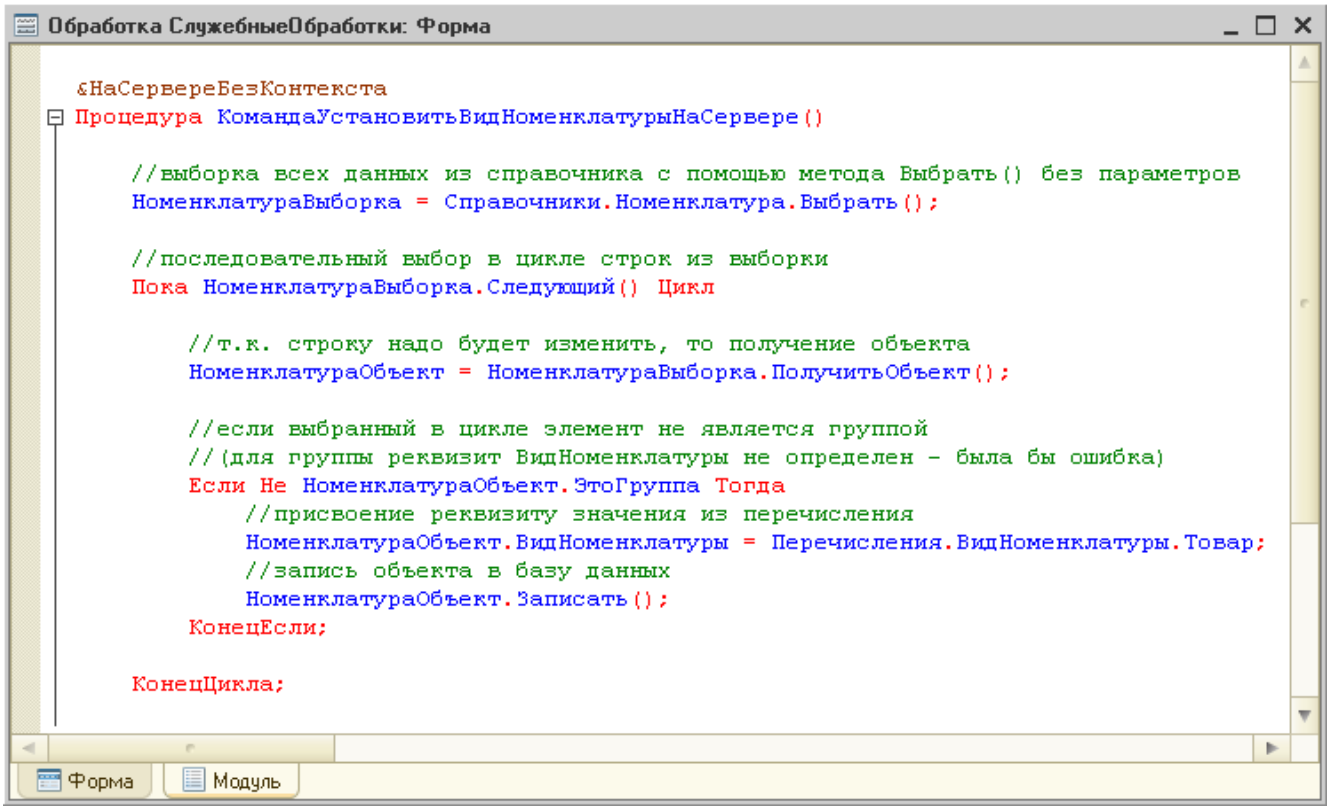


Рис. Программный код для установления реквизита ВидНомеклатуры у всех элементов справочника Номенклатура

Обработчик команды готов, но команда для пользователя пока не доступна. Для того чтобы она появилась на форме в виде кнопки, необходимо переключиться на вкладку Форма и перетащить команду в список элементов формы (рис.).

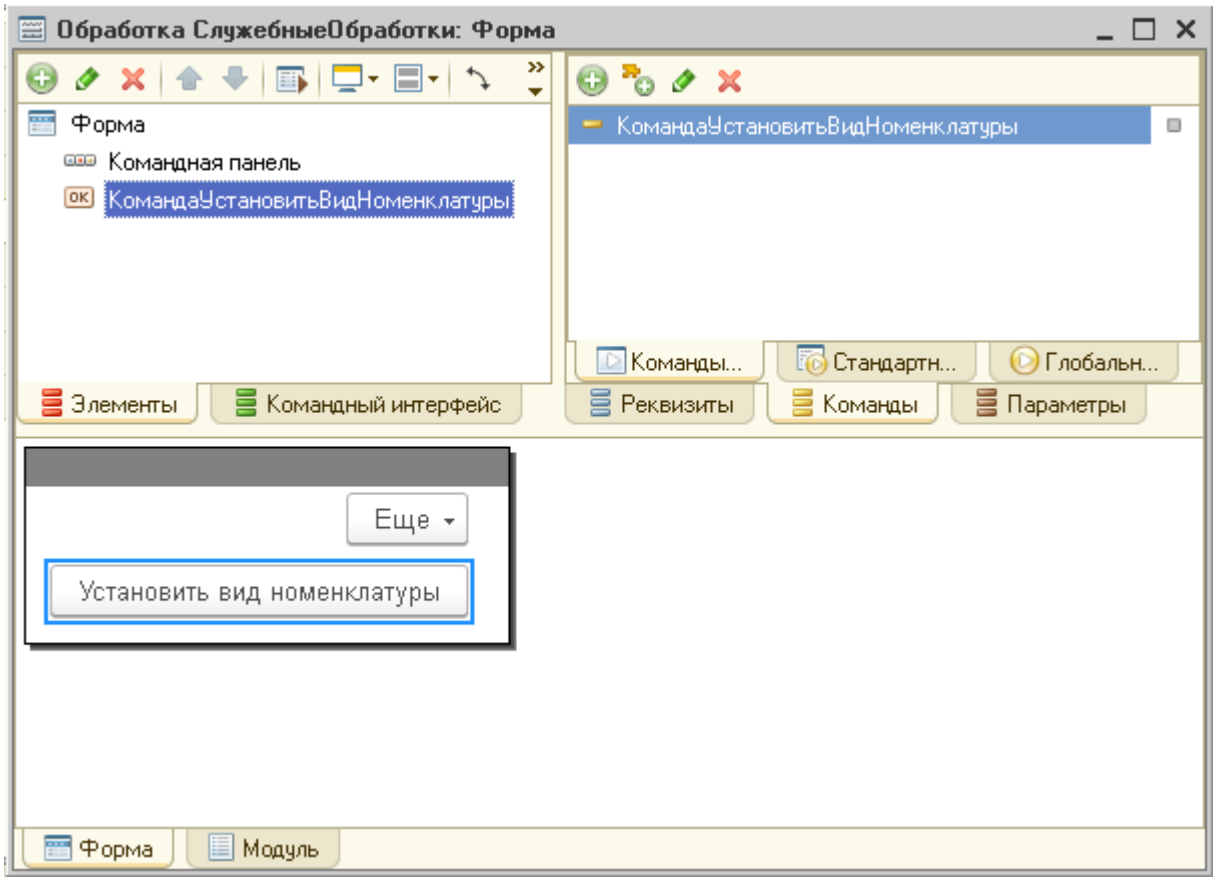


Рис. Добавление команды на форму

Запустим приложение и откроем форму созданной обработки. По умолчанию все команды помещаются в меню Сервис (рис.).

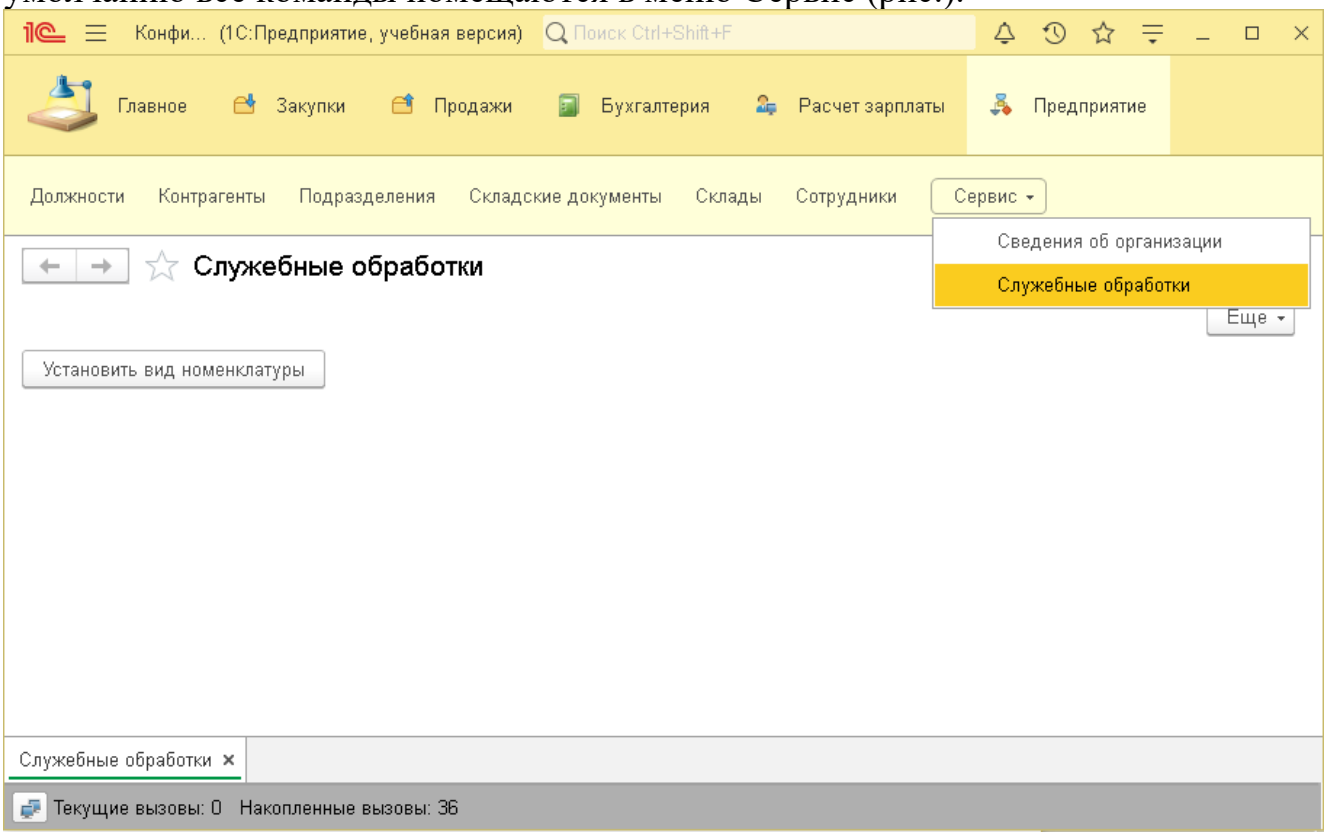


Рис. Форма обработки в приложении

Нажмем на кнопку УстановитьВидНоменклатуры и затем откроем справочник Номенклатура. Для наглядности временно установим режим просмотра Список (рис.).

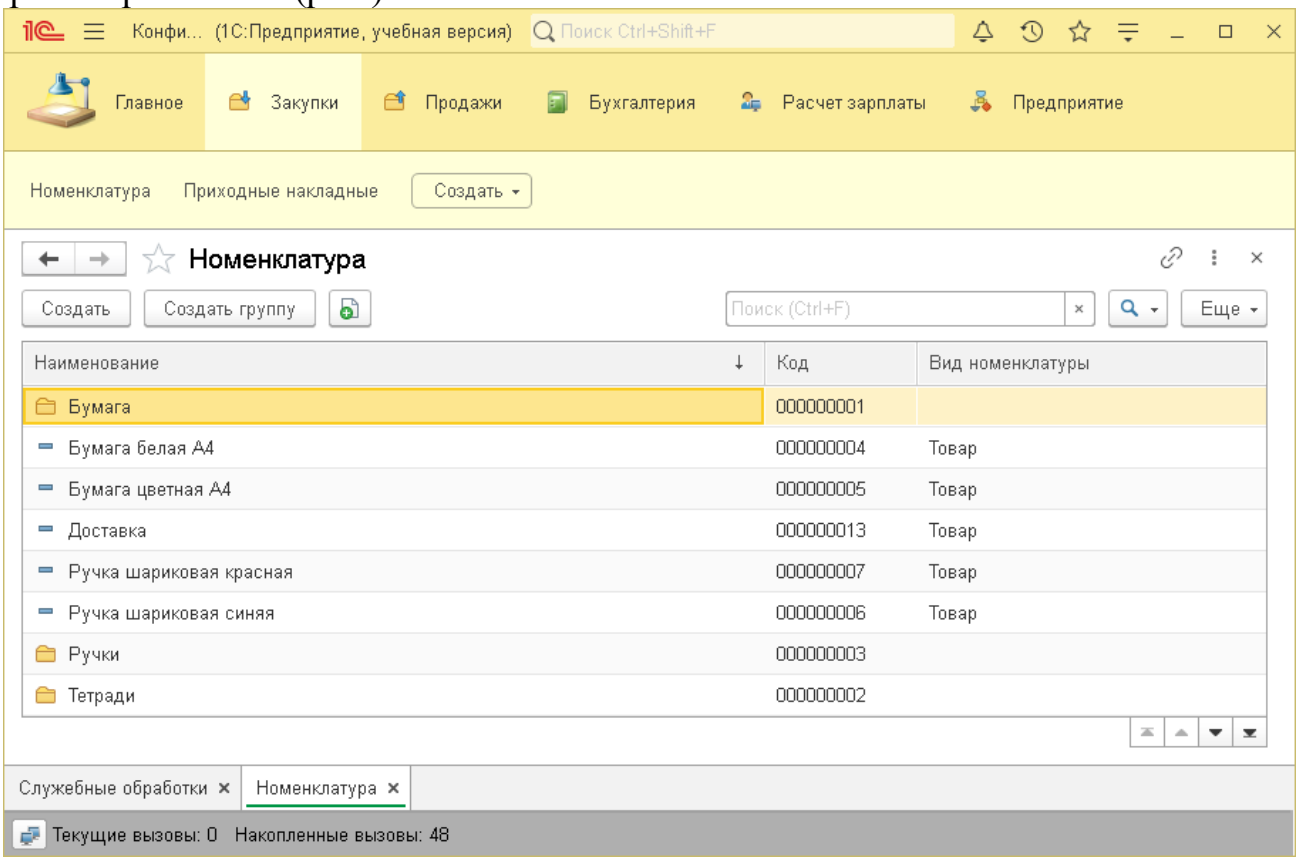


Рис. Форма списка справочника Номенклатура после заполнения реквизита Вид номенклатуры

Как видно, у всех элементов справочника, кроме групп, установлен вид номенклатура Товар. У элемента Доставка вид номенклатуры с Товар на Услуга нужно изменить вручную.

Выполнение запроса с помощью табличной модели

Добавим в обработку новую команду ВыполнитьЗапрос. В процедуре на сервере напишем следующий код (рис.).

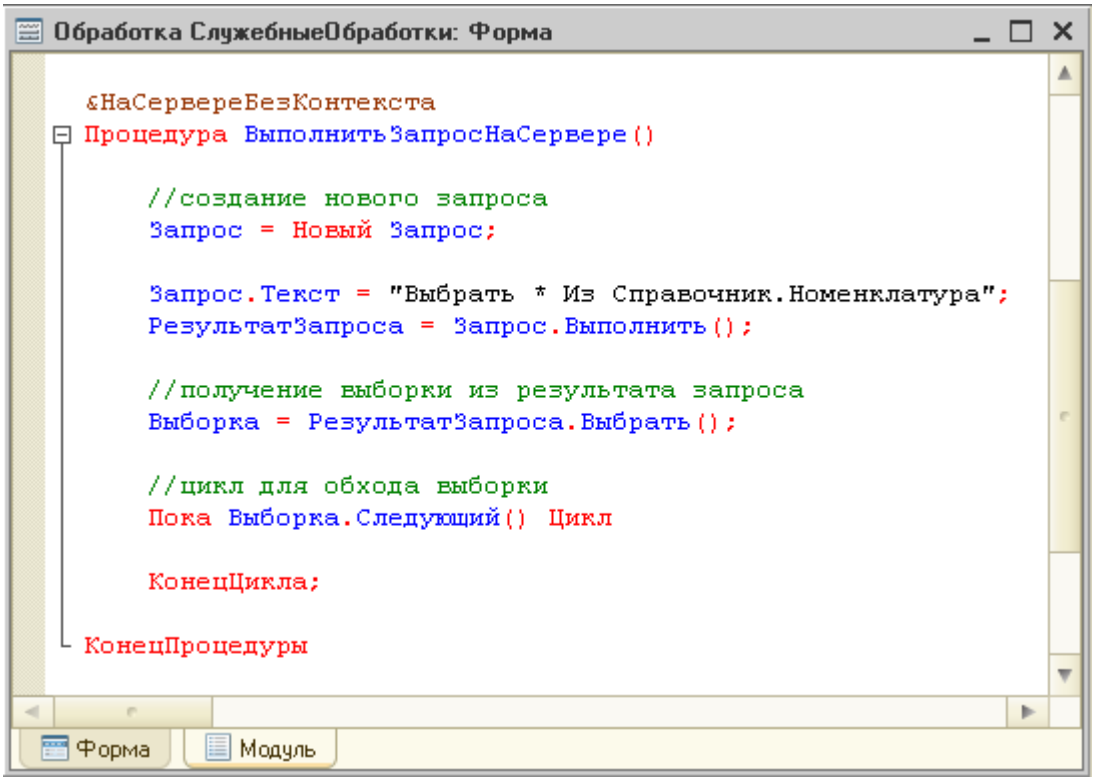


Рис. Программный код запроса к базе данных

Пока результат запроса в приложении никуда не выводится, но его можно будет просмотреть при пошаговой отладке.

Программные модули системы 1С: Предприятие

Текст на встроенном языке программирования располагается в программных модулях 1С. В зависимости от объекта, для описания поведения которого создан модуль, различают следующие виды программных модулей:

- *модуль приложения* относится ко всей конфигурации в целом и может быть только один; предназначен для обработки таких событий приложения, как запуск и завершение работы, а также перехвата внешних событий (например, от оборудования); отслеживает события только при интерактивном запуске приложения пользователем;

- *модуль внешнего соединения* - если конфигурация запускается не в режиме клиентской сессии, а через внешнее программное соединение (например, через COM-соединение), то вместо модуля приложения используется модуль внешнего соединения, который в конфигурации также может быть только один; может содержать обработчики только событий запуска приложения и завершения его работы;

- *модуль сеанса* нужен для того, чтобы инициализировать *параметры сеанса* – специальные объекты конфигурации, предназначенные для хранения определенных параметров для каждого сеанса пользователя на время этого сеанса (например, имени текущего пользователя); в модуле сеанса доступно создание обработчика события УстановкаПараметровСеанса; данный обработчик вызывается самым первым при запуске приложения;

- *общие модули* принадлежат всей конфигурации в целом, но, в отличие от модуля приложения, их может быть несколько;

- *модули объектов* существуют у некоторых прикладных объектов конфигурации (справочников, документов и т.д.);

- *модули форм* - у каждой формы есть модуль, в котором определяется поведение формы и действия, выполняемые из нее, например, открытие других форм.

Правила видимости экспортируемых переменных, процедур и функций различных модулей

В общем модуле недоступны экспортируемые переменные, процедуры и функции модуля приложения (модуля внешнего соединения).

В модуле приложения (модуле внешнего соединения) доступны экспортируемые процедуры и функции общих модулей.

В общих модулях доступны экспортируемые процедуры и функции других общих модулей.

В модулях прикладных объектов и модулях форм доступны экспортируемые переменные, процедуры и функции модуля приложения (модуля внешнего соединения), а также экспортируемые процедуры и функции общих модулей.

Если у формы назначен основной реквизит, то контекст модуля формы содержит дополнительные свойства и методы, связанные с основным реквизитом. Например, в модуле формы элемента справочника Номенклатура доступны свойства и методы объекта СправочникОбъект.Номенклатура.

Структура модуля

В программном модуле в общем случае могут присутствовать следующие разделы в приведенной ниже последовательности:

- раздел объявления переменных;
- раздел описания процедур и функций;
- основной текст программы.

Некоторые разделы могут присутствовать только в модулях определенного вида. Например, обработчики событий элементов форм могут присутствовать только в модулях форм, а раздел описания переменных и раздел инициализации не могут быть определены в неглобальных общих модулях, модулях менеджеров объектов, наборов записей, значений констант и модуле сеанса.

Автоматический пересчет суммы в строках документа

Табличная часть документа Приходная накладная содержит реквизит Сумма. Пока его нужно вводить вручную, что является достаточно неудобным.

Откроем в конфигураторе форму данного документа. В редакторе формы в списке элементов раскроем табличную часть Товары и выделим реквизит

ТоварыКоличество или выделим столбец Количество на самой форме (рис.). Двойным щелчком откроем панель свойств данного реквизита и найдем раздел События (рис.).

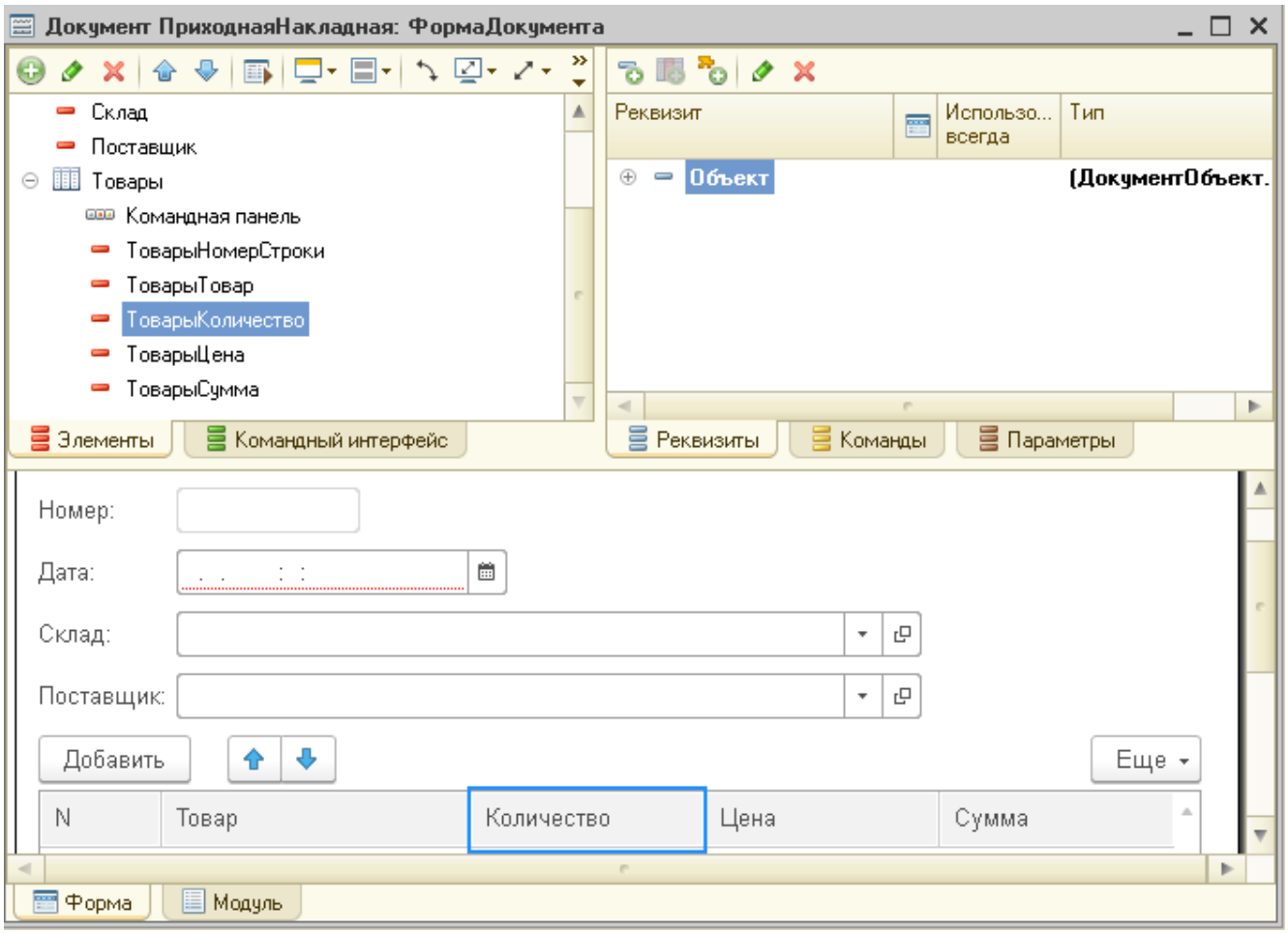


Рис. Конструктор формы документа ПриходнаяНакладная

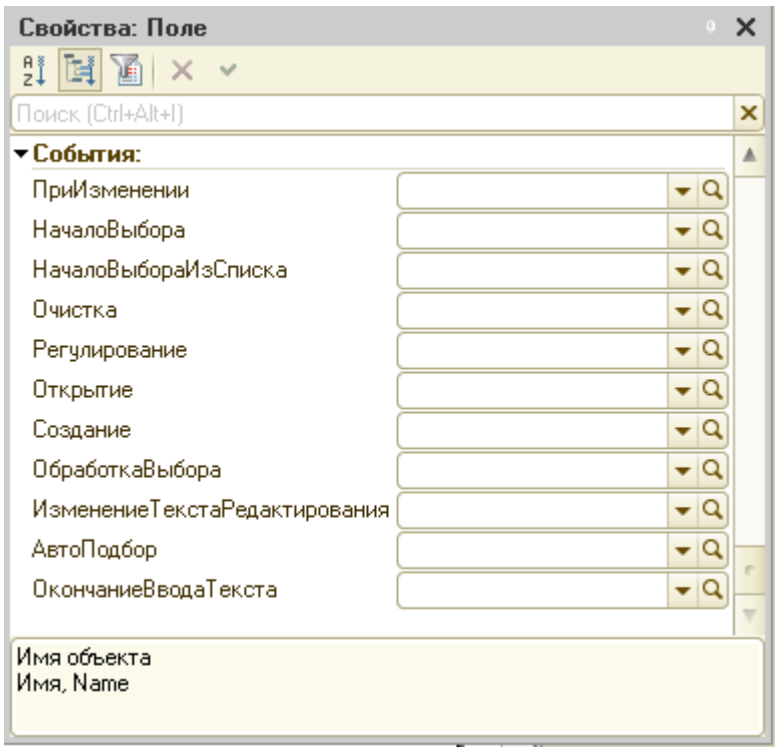


Рис. Список событий формы в панели свойств

Создадим обработчик события ПриИзменении, нажав на пиктограмму Открыть () в поле данного свойства. Система предложит выбрать место исполнения обработчика события (рис.).

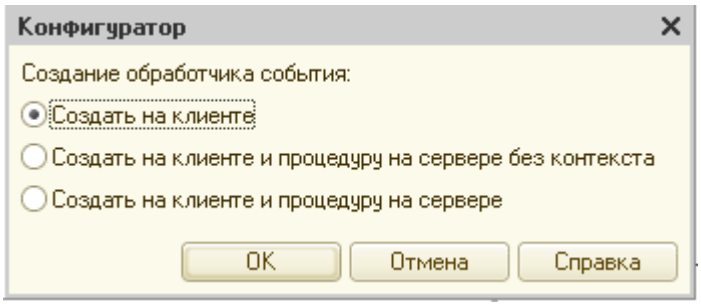


Рис. Окно выбора места исполнения обработчика события

Оставим вариант Создать на клиенте и нажмем кнопку ОК. Откроется вкладка Модуль редактора форм (рис.).

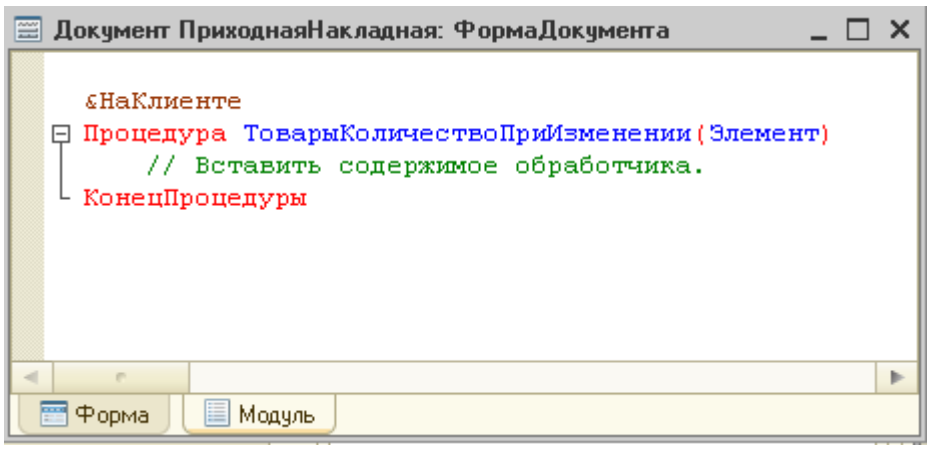


Рис. Шаблон процедуры в модуле формы

Напишем внутри процедуры ТоварыКоличествоПриИзменении() следующий код (рис.).

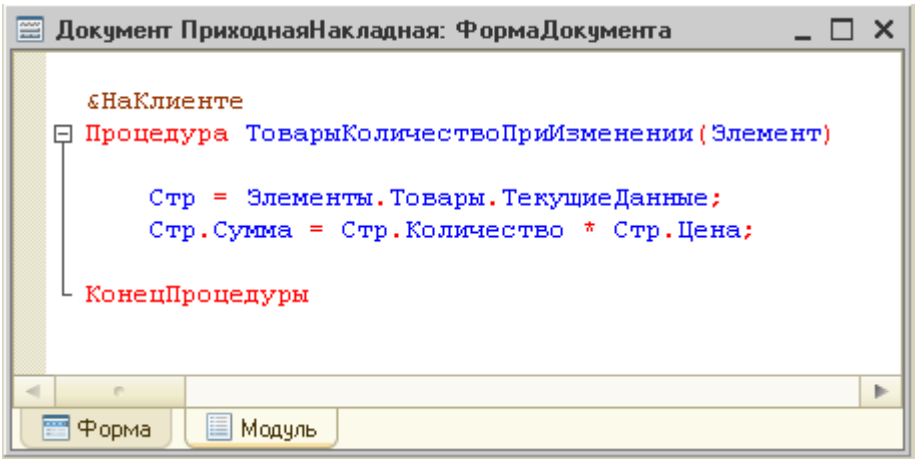


Рис. Процедура расчета суммы в модуле документа

В первой строке в переменную Стр записывается содержимое выбранной пользователем строки табличной части Товары. Объект Элементы содержит все элементы формы. Чтобы обратиться к конкретному элементу, необходимо указать его имя через точку. В данном случае обращаемся к табличной части Товары. Получить выбранную строку в виде структуры можно при помощи свойства ТекущиеДанные. Обратиться к значению столбца структуры можно через точку.

Во второй строке процедуры вычисляется значение столбца Сумма как произведение значений колонок Количество и Цена.

Запустим приложение в режиме отладки и откроем какую-либо приходную накладную. Если теперь попытаться изменить в табличной части количество товара, то сумма рассчитается автоматически.

Поскольку процедура вычисления суммы в приложении может понадобиться неоднократно, то вынесем ее в общий модуль.

Создадим общий модуль. Для этого в дереве объектов конфигурации в ветви Общие выделим объект Общие модули и нажмем кнопку Добавить. Откроется окно модуля. В панели свойств модуля зададим имя РаботаСДокументами.

Поскольку при расчете суммы выполняется работа только с данными формы и нет обращений к базе данных, то можно весь модуль компилировать и исполнять на клиенте. В панели свойств снимем флажок Сервер и установим Клиент (рис.).

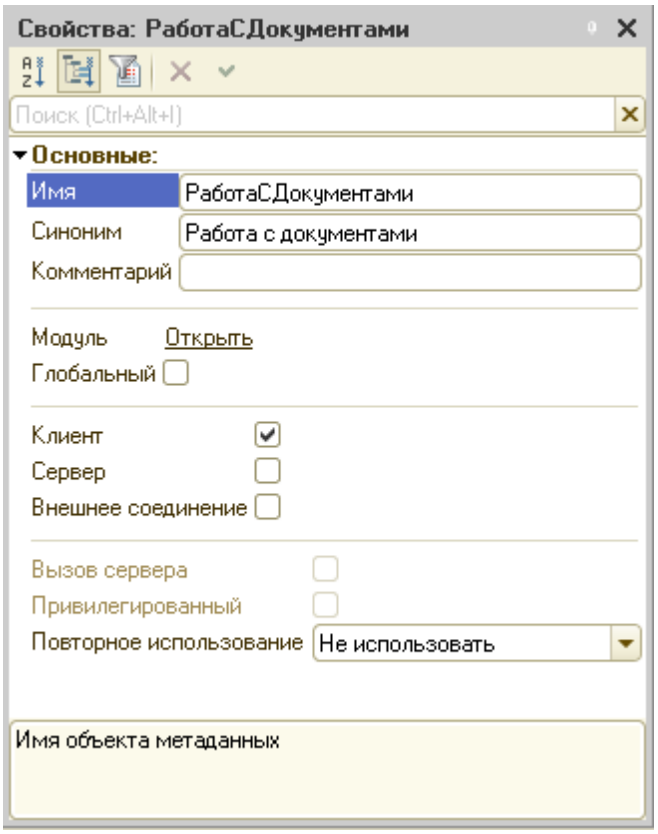


Рис. Панель свойств общего модуля

Добавим в модуль процедуру вычисления суммы (рис.).

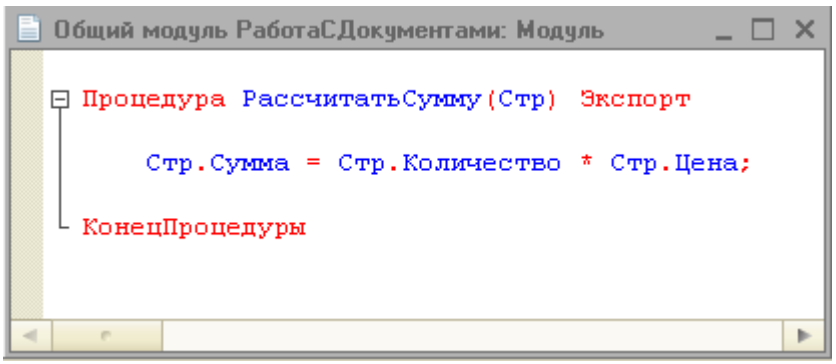


Рис. Процедура расчета суммы в общем модуле

В процедуру РассчитатьСумму передается переменная Стр, которая определяется в процедуре ПриИзменении поля Количества. Ключевое слово Экспорт в заголовке процедуры указывает на то, что эта процедура может быть доступна из других программных модулей.

Отредактируем текст процедуры ТоварыКоличествоПриИзменении() (рис.).

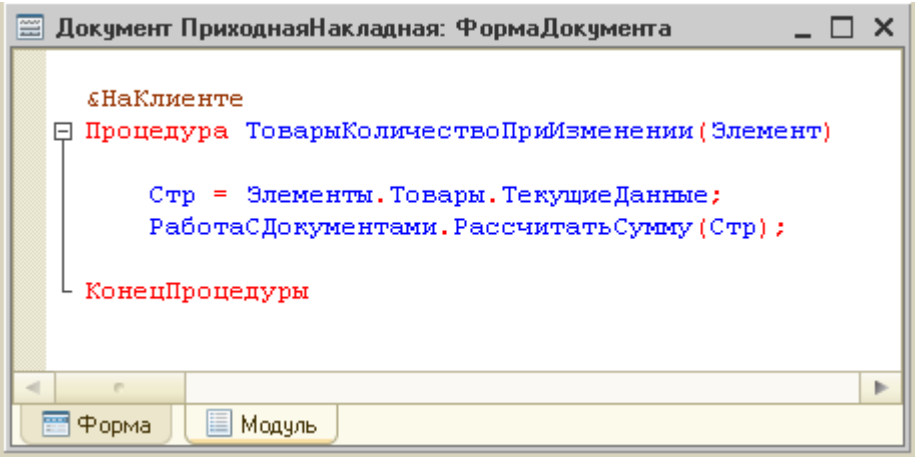


Рис. Вызов процедуры из общего модуля

Первая строка процедуры осталась без изменений, а во второй вместе непосредственного расчета суммы вызывается процедура РассчитатьСумму() из общего модуля РаботаСДокументами и в качестве параметров передается ей текущая строка табличной части Товары.

Анализ кода с помощью отладчика

Для отладки программных модулей в конфигураторе 1С предназначен специальный встроенный инструмент – *отладчик*. Он предоставляет следующие возможности:

- пошаговое выполнение модуля,
- расстановка точек останова,
- прерывание и продолжение выполнения модуля,
- возможность отладки нескольких модулей одновременно,
- вычисление выражений для анализа состояния переменных,
- просмотр стека вызовов процедур и функций, возможность остановки по возникновению ошибки,
- возможность редактирования модуля в процессе отладки.

Все команды отладчика, доступные в режиме разработки приложения, находятся в меню Отладка. Состав команд данного меню меняется в зависимости от активного окна и режима работы системы. Меню Отладка при открытом тексте программного модуля показан на рис.

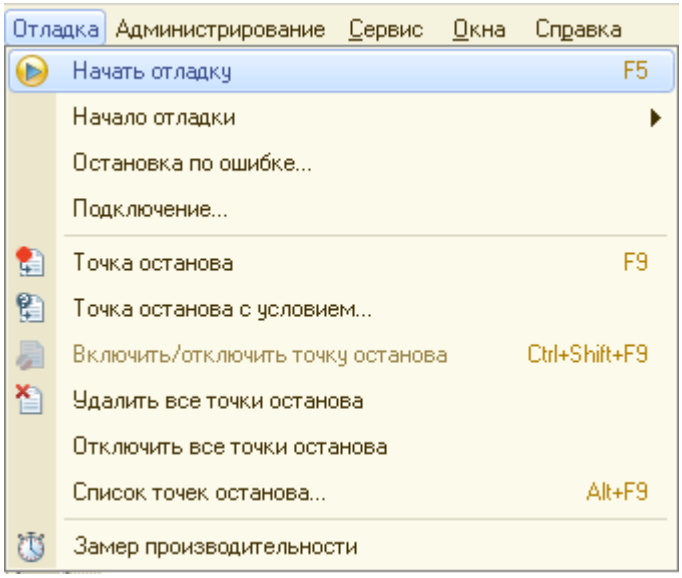


Рис. Меню Отладка

Отладчик позволяет разработчику выполнить пошаговую проверку отдельного фрагмента программного кода во время исполнения приложения. С этой целью используются *точки останова* – специальные маркеры, при достижении которых исполнения программы приостанавливается и управление передается в отладчик.

Команды для работы с точками останова доступны при редактировании текста программного модуля в меню Отладка и на специальной панели инструментов (рис.).

Рис. Панель инструментов Точки останова

Установить / снять точку останова можно следующими способами:

- выбрать команду Отладка / Точка останова;
- нажать команду Точка останова на панели инструментов Точки останова;
- дважды щелкнуть мышью в служебной области рядом с нужной строчкой
- нажать кнопку F9 на клавиатуре.

Для снятия всех точек останова нужно выбрать команду Удалить все точки останова или Отключить все точки останова в меню Отладка или на панели инструментов Точки останова.

Признаком установки точки останова является красный маркер рядом со строкой программного кода.

Рассмотрим работу с отладчиком на примере процедуры ТоварыКоличествоПриИзменении(). Установим точку останова на первой строчке (рис.).

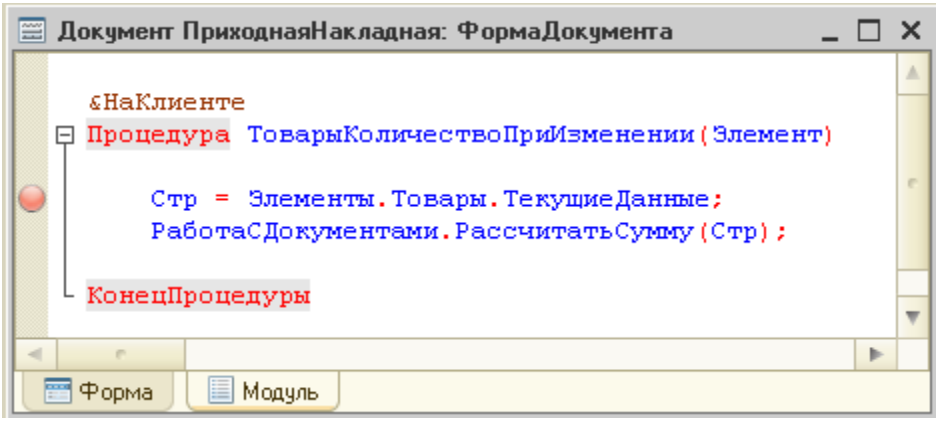


Рис. Установка точки останова

Запустим отладку конфигурации, откроем приходную накладную и изменим поле Количество в любой строке документа. Выполнение программы прервется, и в конфигураторе будет открыта процедура ТоварыКоличествоПриИзменении в точке останова. Значок точки останова поменяется на стрелку ( ). Это означает, что программа остановилась перед исполнением данной строки.

В таком режиме в меню Отладка и на панели инструментов появляются дополнительные команды для работы с конфигурацией в режиме отладки (рис.).

Рис. Панель инструментов Отладка конфигурации

Назначение команд панели инструментов Отладка конфигурации перечислено в табл.

№	Пиктограмма команды	Название	Назначение
1.		Шагнуть в	Если текущая строчка кода содержит вызов процедуры / функции, то переход к вызываемой процедуре, иначе переход к следующей строчке
2.		Шагнуть через	Переход к следующей строке программного кода
3.		Шагнуть из	Переход к функции / процедуре, вызвавшей данную
4.		Идти до курсора	Выполнение программного кода до строки, в которой установлен курсор
5.		Текущая строка	Возврат к строке, которая будет исполняться следующей (если разработчик пролистал текст модуля далеко)
6.		Продолжить отладку	Исполнение программы до следующей точки останова
7.		Вычислить выражение	Вычисление значения переменной
8.		Табло	Специальное окно для просмотра значений переменных
9.		Локальные	Специальное окно, содержащее

		переменные	список значений локальных переменных процедуры
10.		Стек вызовов	Содержит последовательность вызова процедур и функций в процессе исполнения программы до текущей строки
11.		Замер производительности	Вспомогательное окно, показывающее время исполнения отдельных фрагментов программного кода

Нажмем кнопку Шагнуть через. После этого выполнится первая строка процедуры, следовательно, переменной Стр будет присвоено какое-то значение. Откроем Табло и перетащим в него переменную Стр (рис.).

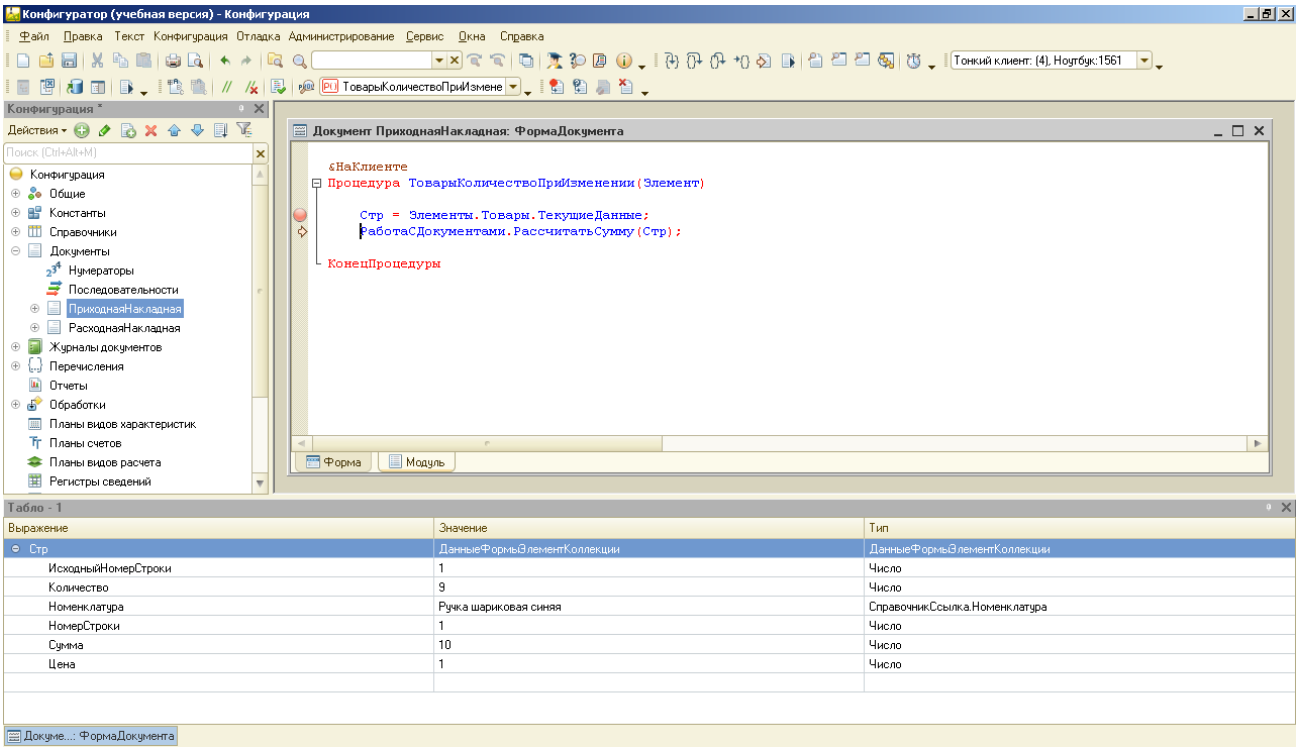


Рис. Просмотр значение переменной в процессе отладки

Нажмем на пиктограмму команды Шагнуть в. После этого откроется модуль Работа с документами. Откроем окно Стек вызовов для просмотра последовательности вызова процедур (рис.).

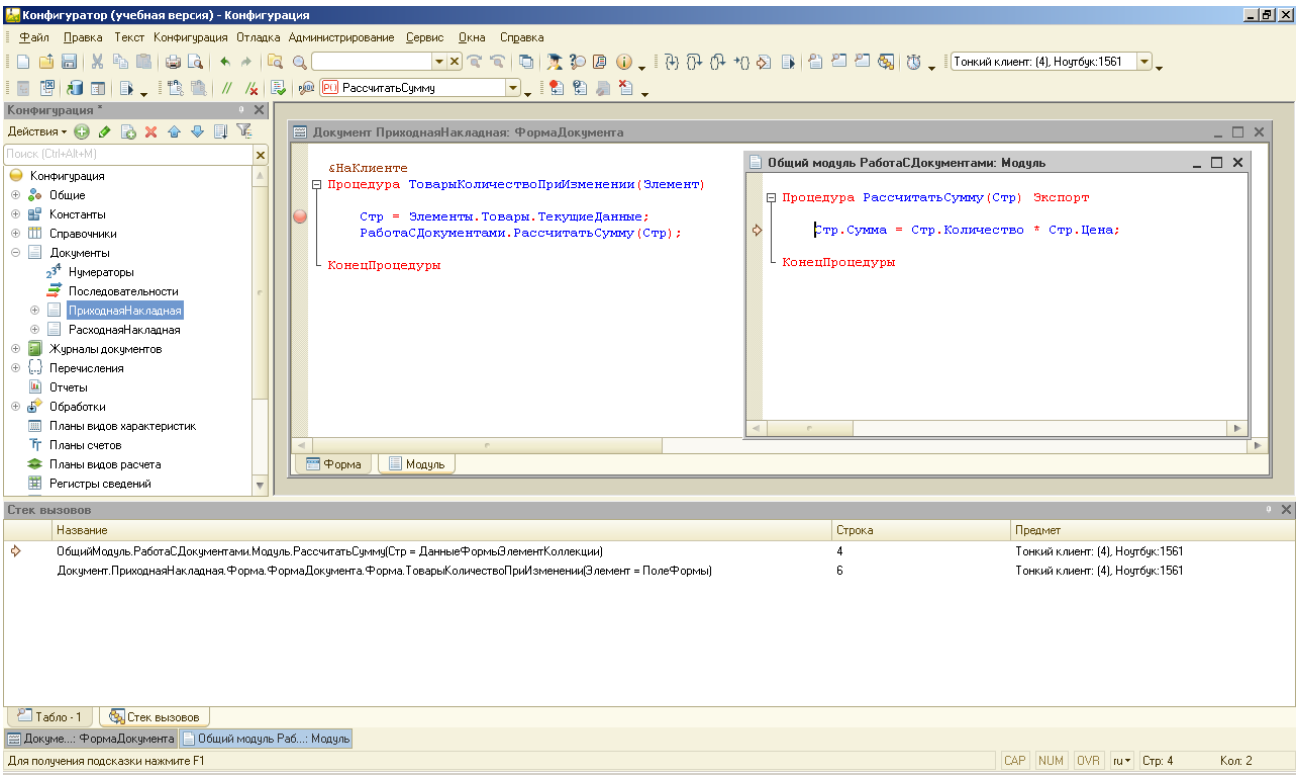


Рис. Использование стека вызовов для отладки конфигурации

Завершить отладку приложения можно с помощью команды Отладка / Завершить.

Запустим пошаговую отладку для процедуры ВыполнитьЗапросНаСервере() обработки и выведем в табло доступные результаты (рис.).

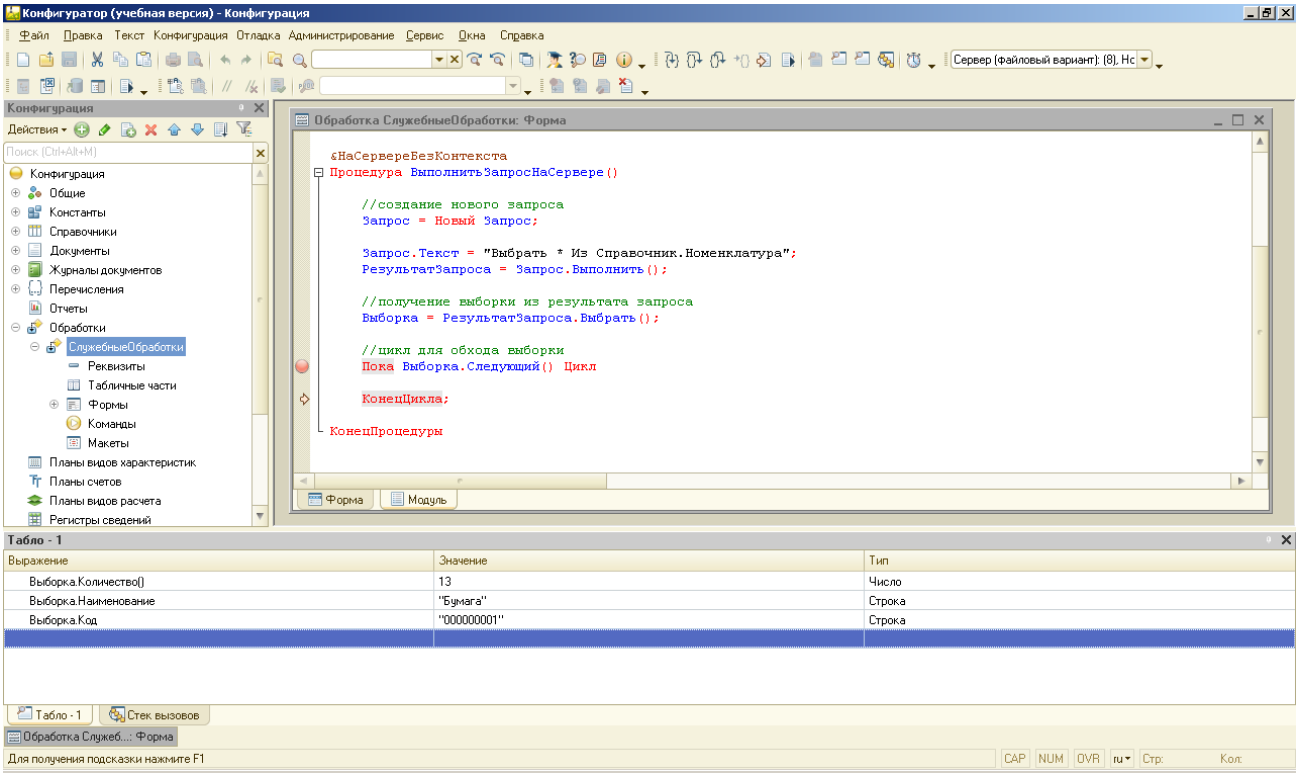


Рис. Отладка процедуры ВыполнитьЗапросНаСервере

Как видно, в табло можно просмотреть только отдельные свойства выборки (например, Количество()) или данные текущего элемента (Код, Наименование).

Для того чтобы просмотреть в отладчике весь результат запроса, выгрузим его в таблицу значений (рис.).

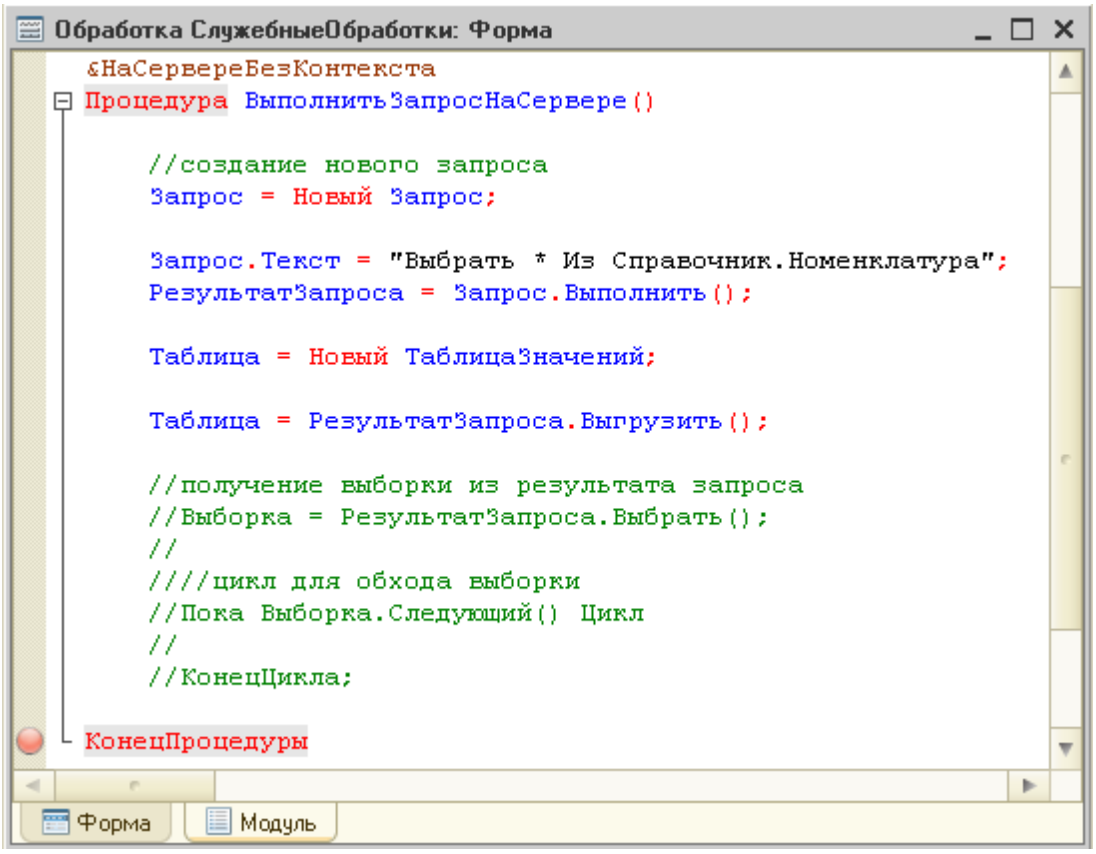


Рис. Выгрузка результата запроса в таблицу значений

Установим точку останова в конце процедуры и запустим отладку. Выделим название переменной Таблица и в панели инструментов Отладка выберем команду Вычислить выражение. В открывшемся окне нажмем пиктограмму Показать значение в отдельном окне (рис.).

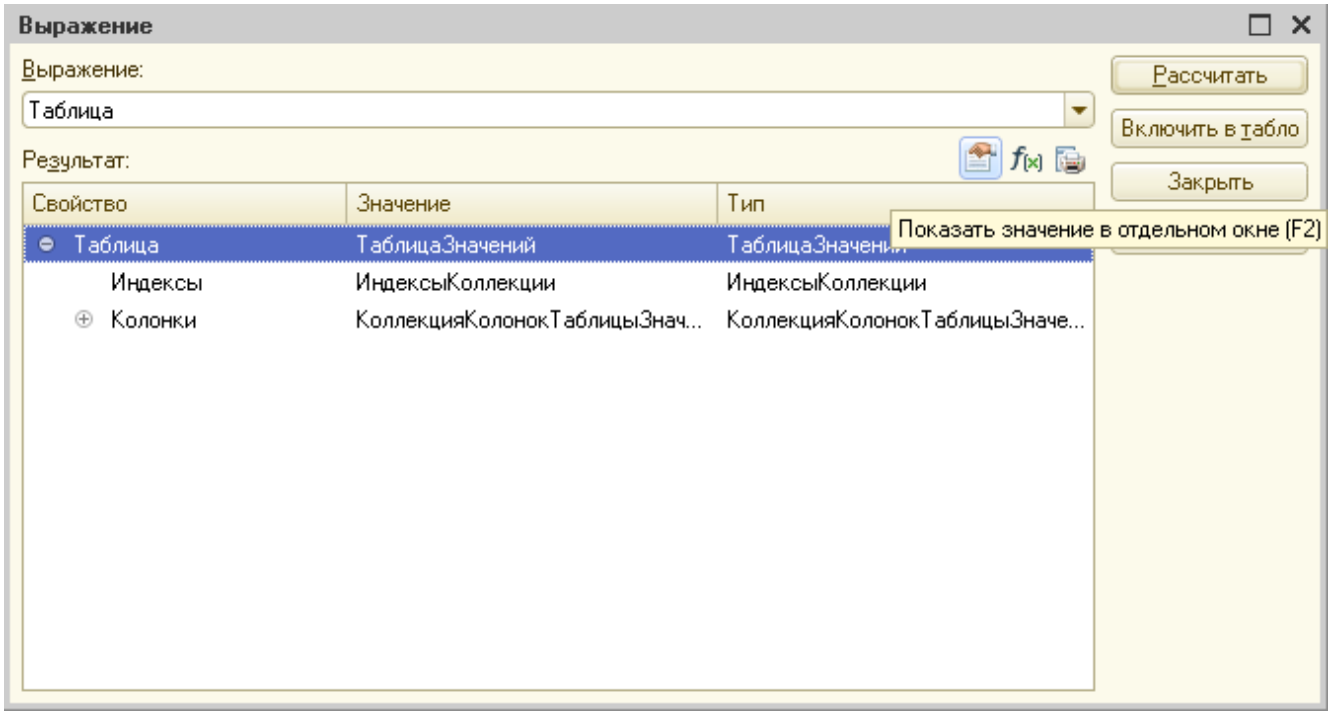
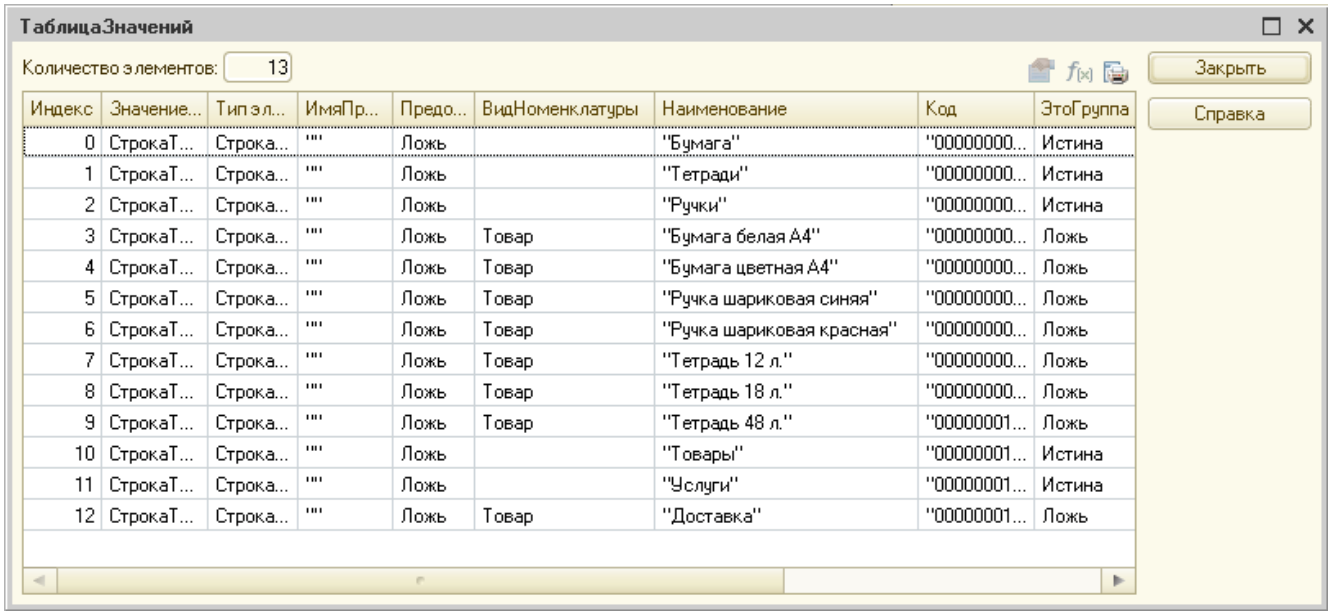


Рис. Диалоговое окно Выражение

В следующем окне будет выведена вся информация из таблицы значений (рис.).



Индекс	Значение...	Тип эл...	ИмяПр...	Предо...	ВидНоменклатуры	Наименование	Код	ЭтоГруппа
0	СтрокаТ...	Строка...	""	Ложь		"Бумага"	"00000000...	Истина
1	СтрокаТ...	Строка...	""	Ложь		"Тетради"	"00000000...	Истина
2	СтрокаТ...	Строка...	""	Ложь		"Ручки"	"00000000...	Истина
3	СтрокаТ...	Строка...	""	Ложь	Товар	"Бумага белая А4"	"00000000...	Ложь
4	СтрокаТ...	Строка...	""	Ложь	Товар	"Бумага цветная А4"	"00000000...	Ложь
5	СтрокаТ...	Строка...	""	Ложь	Товар	"Ручка шариковая синяя"	"00000000...	Ложь
6	СтрокаТ...	Строка...	""	Ложь	Товар	"Ручка шариковая красная"	"00000000...	Ложь
7	СтрокаТ...	Строка...	""	Ложь	Товар	"Тетрадь 12 л."	"00000000...	Ложь
8	СтрокаТ...	Строка...	""	Ложь	Товар	"Тетрадь 18 л."	"00000000...	Ложь
9	СтрокаТ...	Строка...	""	Ложь	Товар	"Тетрадь 48 л."	"00000001...	Ложь
10	СтрокаТ...	Строка...	""	Ложь		"Товары"	"00000001...	Истина
11	СтрокаТ...	Строка...	""	Ложь		"Услуги"	"00000001...	Истина
12	СтрокаТ...	Строка...	""	Ложь	Товар	"Доставка"	"00000001...	Ложь

Рис. Таблица значений с результатами запроса к базе данных

Задание на лабораторную работу:

1. Выполнить все примеры, рассмотренные в лабораторной работе.
2. Изменить процедуру команды УстановитьВидНоменклатуры таким образом, чтобы вид устанавливался в зависимости от названия группы, в которой находится элемент.

Замечания:

Группа, в которой находится элемент, хранится в его свойстве Родитель. Чтобы найти группу на самом верхнем уровне иерархии (в примере это две группы «Товары» и «Услуги»), необходимо в цикле перебирать родителей элементов до тех пора, пока поле Родитель не будет пустым, то есть не будет равно значению Неопределено.

3. Для документа Приходная накладная реализовать пересчет суммы при изменении цены товара.

4. Для документа Расходная накладная реализовать автоматический пересчет суммы при изменении количества товара и цены.

5. Написать запросы к другим справочникам и документам приложения и вывести их содержимое в отладчике.

6. Выгрузить информационную базу и сохранить ее для следующей лабораторной работы.