

Лабораторная работа № 2

Форма документа, программирование форм, события формы.

Общие сведения

1. Форма документа

Когда мы открываем справочник или документ, программа сама создает стандартное представление для этого объекта. При этом она анализирует довольно много условий: необходимость использования стандартных команд, вид объекта, структуру (дополнительные реквизиты и табличные части), роли, функциональные опции и т.д.

Тем не менее, формы получаются не такие, какими мы их хотели бы их видеть. Если мы хотим открыть объект в собственном окне, его надо создать и называться такое окно будет «Форма». Как правило, формы «привязаны» к конкретному объекту: справочник «Номенклатура», документ «Поступление товаров». Но бывают независимые формы, которые могут отображать данные нескольких объектов или использовать как-то иначе. Формы объектов открываются стандартными командами глобального командного интерфейса конфигурации, произвольные формы могут открываться программно.

Форма отображает данные объекта информационной базы, актуальные на момент открытия. Если эти данные были изменены, например, другим пользователем, их можно «пересчитать». Пункт «Пересчитать» доступен пользователю в меню «Все действия» формы, документа или элемента справочника.

Сначала надо создать форму. Делаем мы это на вкладке «Формы» свойств документа. Если в свойствах конфигуратора включен режим «Управляемое и обычное приложение», будут видны две вкладки «Основные» и «Дополнительные», которые позволяют нарисовать для одного объекта два независимых комплекта форм (управляемые и обычные). Если вкладок нет, значит, все формы будут управляемые по умолчанию (рис. 1).

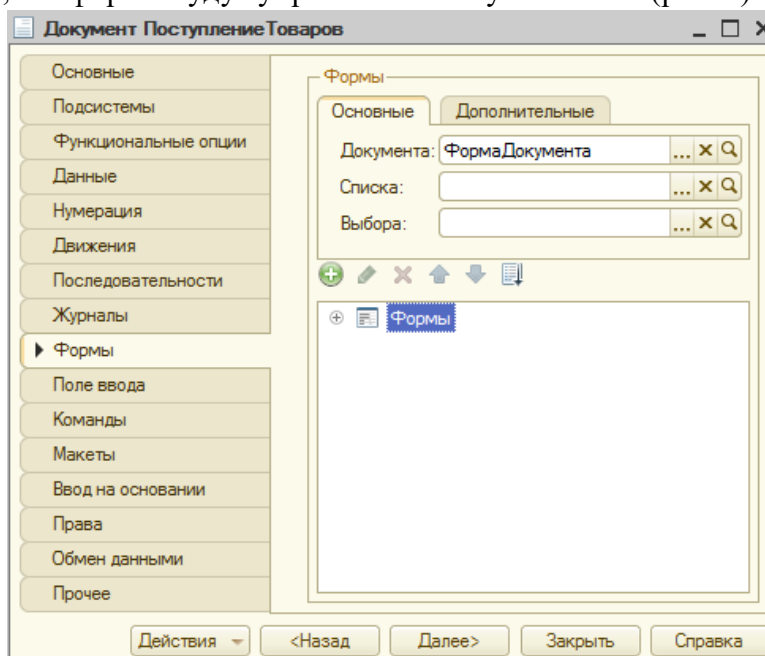


Рис. 1 Вкладка «Формы» документа

Здесь можно создать для одного документа неограниченное число форм, но только три из них (форма документа, форма списка, форма выбора) платформа сама знает, как использовать, когда открывать. Все остальные, которые вы создадите, придется открывать программно.

- Форма документа, в которой будут просматриваться и редактироваться данные одного экземпляра документа.
- Форма списка, для просмотра списка документов.
- Форма выбора, которая будет отображать список документов для выбора ссылки на документ (например, документ-основание).

На вкладке «Основные» по нажатию «лупы» справа от формы документа, создаем новую форму документа. Запускается конструктор формы объекта (рис. 2).

Конструктор формы документа

Выберите тип формы:

- ☒ Форма документа
- ☐ Форма списка документа
- ☐ Форма выбора документа
- ☐ Произвольная форма

Тип формы: Управляемая

☐ Назначить форму основной

☐ Основная форма списка и выбора

Имя:

Синоним:

Комментарий:

☒ Командная панель формы сверху

☐ Командная панель формы снизу

< Назад Далее > Готово Отмена Справка

Рис.2 Конструктор формы

Нажимаем «Далее», соглашаемся со всем, что предложит конструктор, и «Готово» (рис. 3).

Документ Поступление Товаров: ФормаДокумента1

Форма

- Командная панель
- Номер
- Дата
- Организация
- Контрагент
- Склад

Элементы Командный интерфейс

Реквизит	Использовать всегда	Тип
Объект		(ДокументОбъект.Поступление...
Ссылка	☒	ДокументСсылка.Поступление Товаров
Номер	☐	Строка
Дата	☐	Дата
Проведен	☒	Булево

Реквизиты Команды Параметры

Провести и закрыть Записать Провести Еще ▾

Номер:

Дата:

Организация:

Контрагент:

Склад:

Комментарий:

Форма Модуль

Рис.3 Редактор управляемой формы

Рассмотрим основные инструменты редактора форм и свойства формы.

Левый верхний список – «Элементы» - показывает порядок и группировку элементов управления в диалоге формы.

В правом верхнем углу три закладки. Первая из них – «Реквизиты», которая показывает основной реквизит формы (от выделен жирным списком), и дополнительные реквизиты формы. Основной реквизит формы определяет принадлежность этой формы к какому-то объекту конфигурации, именно его данными она будет управлять по умолчанию.

2. Объектная модель

1С:Предприятие имеет три модели представления своих данных — объектную, табличную и xml. Все три модели реализованы непосредственно в платформе 1С. Это важно учитывать при разработке архитектуры конфигурации, от этого зависит быстродействие прикладного решения.

Объектная модель данных 1С – доступ к данным с помощью встроенного языка, представляя базу данных в виде набора объектов, имеющих свои свойства и методы.

Например, мы ввели элемент справочника «Лопата». Данный элемент для нас является цельным объектом. Это значит, что у него есть:

- Данные этого элемента (заполненный пользователем список полей)
- Стандартный «методы» объекта
- Не стандартные «методы» объекта (написанные программистом)
- Ссылка (Уникальный «номер» конкретного элемента справочника).

Все перечисленные свойства объекта составляют вместе Объект 1С. С таким объектом можно работать из программы 1С.

Свойство	Значение	Тип
Номенклатура	Стол обеденный	СправочникСсылка.Номенклатура
БазоваяЕдиницаИзмерения	шт	СправочникСсылка.КлассификаторЕдиницИзмерения
ВерсияДанных	"AAAAAQAAII"	Строка
Владелец		Неопределено
Код	"00000000134"	Строка
Комментарий	""	Строка
Наименование	"Стол обеденный"	Строка
НаименованиеПолное	"Стол обеденный"	Строка
НоменклатурнаяГруппа		СправочникСсылка.НоменклатурныеГруппы
НомерГТД		СправочникСсылка.НомерГТД
ОсновнаяСпецификацияНоменклатуры		СправочникСсылка.СпецификацияНоменклатуры
ПометкаУдаления	Ложь	Булево
Предопределенный	Ложь	Булево
Родитель	Мебель	СправочникСсылка.Номенклатура
Ссылка	Стол обеденный	СправочникСсылка.Номенклатура
СтавкаНДС	18%	ПеречислениеСсылка.СтавкиНДС
СтатьяЗатрат		СправочникСсылка.СтатьиЗатрат
СтранаПроисхождения		СправочникСсылка.КлассификаторСтранМира
Услуга	Ложь	Булево
ЭтаГруппа	Ложь	Булево

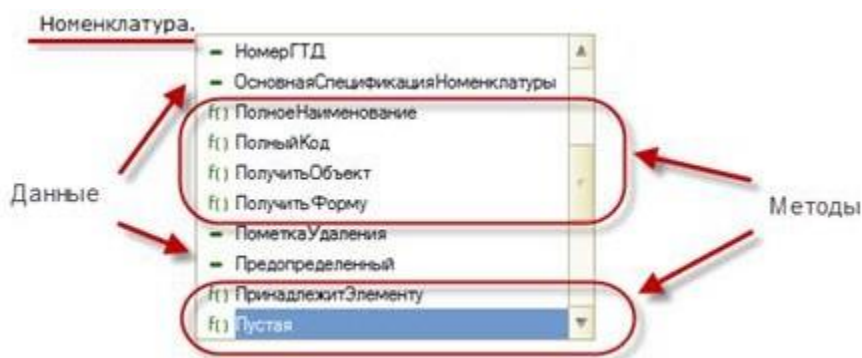


Рис.4 Свойства и методы объекта

Например:

СправочникОбъект.Наименование = «Лопата»;

[СправочникОбъект.Записать\(\)](#);

У каждого объекта 1С есть уникальный идентификатор («номер»). Он всегда уникальный и не может задвоиться. Поле объекта, который содержит идентификатор, называется «Ссылка». Если это поле пустое, значит объект еще не записан в базу данных, иначе оно всегда заполнено.

2.1 Модули

Места для написания программного кода располагаются с тех «уголках» конфигуратора, где, по мнению разработчиков платформы, может быть недостаточно декларативного описания, т.е. там, где мышка и «флажки» скорее всего не справляется с описанием алгоритма работы программы.

Модули есть у форм объектов, для того, чтобы описать логику работы формы (рис.5).

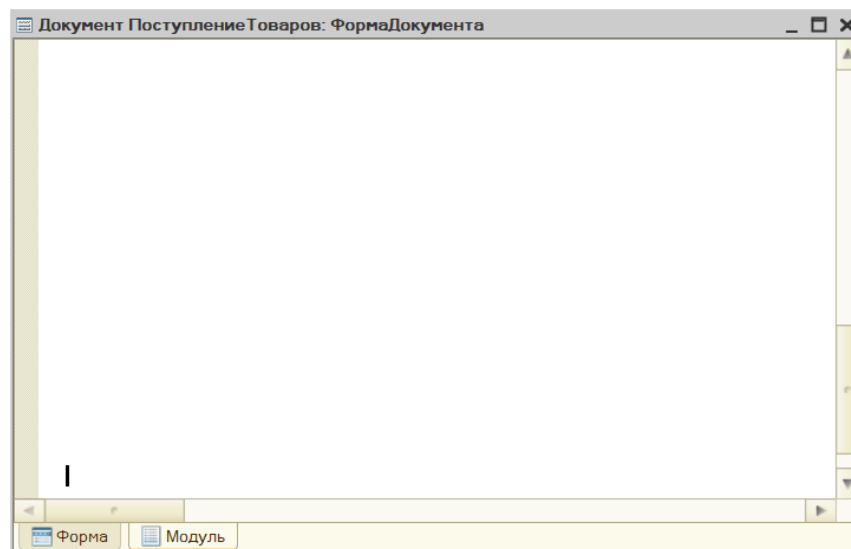


Рис.5 Модуль формы объекта документ

Модуль есть у самого объекта и у менеджера объекта, чтобы описать логику работы документа объекта и всего вида документов на вкладке «Прочее» (рис.6).

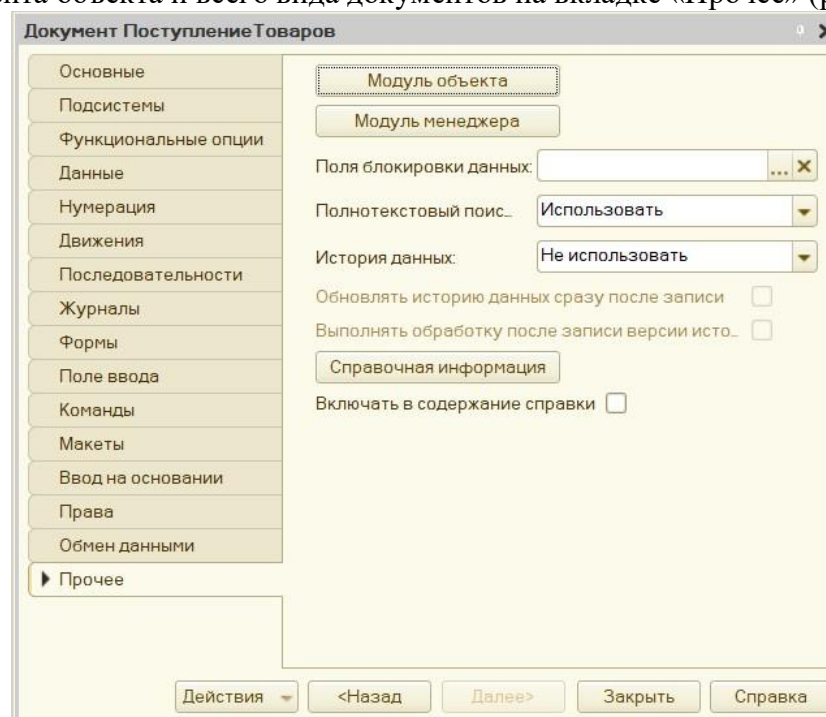


Рис.6 Закладка «Прочее» документа

Модули располагаются во многих других местах, часть из которых мы увидим.

Практически любой модуль разделяется на три части. Кстати, что такое модуль? Это программа, как правило, к чему-то прикрепленная, которая будет выполняться в момент работы с тем, к чему она прикреплена. Например, модуль формы будет выполняться в момент работы пользователя с формой. Часть его выполнится сразу, в момент открытия формы пользователем, часть – по мере вызова.

Алгоритмы необходимо размещать в модулях в строго определенном порядке. Модуль состоит из трех последовательных блоков (областей) (Рис.7).

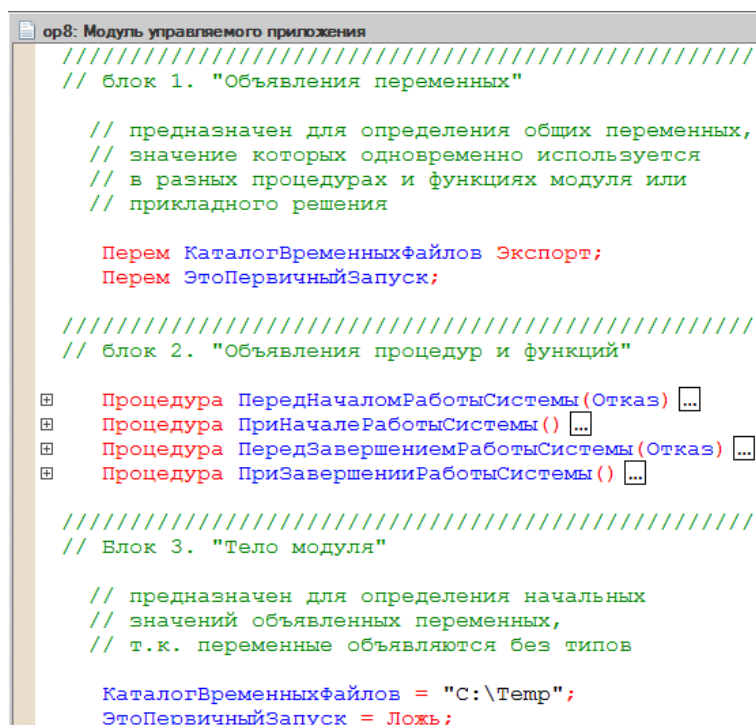


Рис.7 Пример модуля

В некоторых видах модулей могут использоваться не все области размещения алгоритмов.

2.2 Конструкции языка

Как и любой язык, язык программирования имеет свой синтаксис.

Особенности языка программирования

- все «слова» этого языка пишутся слитно без пробелов. Если пробел есть, это уже не одно слово, а два.

- часть слов этого языка зарезервировано и может использоваться только по одному предназначению. Эти слова при стандартной настройке оформления текста модуля выделяются красным цветом шрифта.

- все строки кода заканчиваются точкой с запятой – «;», кроме «КонецПроцедуры», «КонецФункции» и т.д.

- строки кода, которые не должны выполняться, а служат для того лишь, чтобы объяснить предназначение алгоритма – комментарии, начинаются с «//».

Основные конструкции языка:

1) Процедура

Процедура – это «кусочек» программного кода, который будет выполняться не при запуске всего модуля, а тогда, когда его «вызовут». Вызвать процедуру можно из тела модуля, из другой процедуры, просто по кнопке в диалоге формы или при обработке какого-то события.

Процедура <Имя_процедуры>()

// Операторы;

...

КонецПроцедуры

Если для работы процедуре нужны данные, отсутствующие в контексте ее выполнения, эти данные можно передать ей на параметры (в скобки). Имена параметров мы придумываем сами, имена неважны с точки зрения вызова этой процедуры и передачи на них каких-то значений. Важно лишь их количество и порядок следования. Вызываем процедуру по имени с перечнем параметров в скобках.

2) Функция

Функция, как и процедура, - это «кусочек» кода, выполняющийся, когда функцию вызывают. Отличие ее от процедуры только в том, что в результате своей работы она возвращает вычисляемое значение.

Функция <Имя_функции>()

...

Возврат <Возвращаемое значение>;

...

КонецФункции

3) Условный оператор

Как и в любом языке программирования, во встроенном языке 1С 8 присутствует условный оператор. Он позволяет выполнять определенный набор операторов, в зависимости от выполнения логического выражения. Использование условия в 1с позволяет организовать ветвление алгоритма, в зависимости от различных параметров работы системы.

Если <Логическое выражение> Тогда

// Операторы

[ИначеЕсли <Логическое выражение> Тогда]

// Операторы

[Иначе]

// Операторы

КонецЕсли;

Из приведенного синтаксиса, можно сказать об основных особенностях оператора:

- Логический оператор всегда начинается с ключевой фразы *Если* и заканчивается фразой *КонецЕсли*;
- *Логическое выражение* — выражение, результатом которого является значение типа *Булево*, то есть *Истина* или *Ложь*. Если результат выполнения логического выражения — *Истина*, выполняется блок операторов, идущих после ключевого слова *Тогда*;
- В логическом выражении могут быть использованы операции сравнения: «=», «<>» (не равно), «>», «<», «>=» (больше или равно), «<=» (меньше или равно);
- Также в логическом выражении могут применяться булевы операции: «И», «ИЛИ», «НЕ»;

- Можно создавать неограниченное количество блоков *ИначеЕсли*, выражение которых будет проверяться, если выражения предыдущих блоков *Если* и *ИначеЕсли* вернули *Ложь*;
- Также можно использовать один блок *Иначе*. Он будет выполняться, если не выполнен ни один из предыдущих блоков *Если* и *ИначеЕсли*.

4) Циклы

Цикл в 1с предназначен для многократного повторения операторов, находящихся внутри него. В 1с существует три вида циклов:

Вид цикла	Описание
Пока	Цикл осуществляющий повторения, пока выполняется условие.
Для	Цикл осуществляющий заданное количество повторений.
Для каждого	Цикл для обхода коллекций. Обходит каждую строку заданной коллекции.

Цикл Пока

Данный цикл предназначен для осуществления повторений, пока выполняется условие. Синтаксис цикла выглядит так:

```
Пока <Логическое выражение> Цикл
    // Операторы
КонецЦикла;
```

Для выполнения очередного повторения Логическое выражение должно возвращать значение Истина. Это работает следующим образом:

1. Вычисляем значение Логического выражения. Если оно Ложь, цикл завершается. Если Истина:
2. Выполняем операторы цикла;
3. Возвращаемся на п. 1.

Цикл Для

Предназначен для осуществления заданного количества повторений, это так называемый цикл со счетчиком.

```
Для <Счетчик> = <Выражение1> По <Выражение2> Цикл
    // Операторы
КонецЦикла;
```

При каждом проходе цикл увеличивает значение в переменной Счетчик на 1. Начальное значение счетчика указывается в Выражение1, конечное в Выражение2.

Цикл Для Каждого

Используется для обхода коллекций. Может обходить таблицы значений, массивы, структуры, списки значений (и любые другие коллекции).

```
Для каждого <Переменная1> Из <Переменная2> Цикл
    // Операторы
КонецЦикла;
```

В Переменная2 указывается коллекция, а в Переменная1 на каждом проходе цикла будет содержаться элемент коллекции. Контролировать количество элементов при этом не нужно, это делается автоматически. То есть цикл закончит выполняться, как только закончатся элементы коллекции.

2.3 Директивы компиляции модуля

Начиная с платформы 8.2 (далее управляемое приложение), разработчикам стала доступна возможность самим определять, где будет выполняться код, на сервере или на клиенте. Такая необходимость возникла в связи с появлением "тонкого клиента" и "веб-клиента", которые могут работать через интернет, что несомненно накладывает определенные ограничения на передачу данных.

Директивы компиляции, что же это такое? Давайте сначала обозначим, что форма - это объект, который создается в процессе работы прикладного решения и состоит из двух частей, которые выполняются в двух контекстах, на клиенте и на сервере. Взаимодействие между частями осуществляется с помощью серверного вызова.

Директивы компиляции - это специальные ключевые слова, позволяющие указать платформе, в каком контексте необходимо выполнить процедуру или функцию. Указывать директивы для процедур и функций можно не только в модулях форм, но и в модулях команд, а также общих модулях, однако использовать их рекомендуется только в модулях управляемых форм и в модулях команд.

Существует пять различных директив:

&НаКлиенте

&НаСервере

&НаСервереБезКонтекста

&НаКлиентеНаСервереБезКонтекста

&НаКлиентеНаСервере

Директива *&НаКлиенте* указывает на то, что код функций/процедур будет выполняться в контексте клиентского приложения, где будут доступны только элементы, параметры и реквизиты формы. Доступны вызовы любых процедур модуля формы.

При указании директивы *&НаСервере*, для процедур/функций будут доступны не только данные формы, но и возможность обращаться к данным базы. Выполнение кода будет происходить на сервере. Обращение к процедурам на клиенте уже не доступно. Т.е., находясь на сервере, нельзя инициировать вызов клиентских процедур и функций.

Если указана директива *&НаСервереБезКонтекста*, это означает, что код будет выполняться на сервере, но доступа к контексту формы (реквизиты, параметры, элементы) уже не будет. Доступны вызовы только внеконтекстных процедур и функций.

Директива *&НаКлиентеНаСервереБезКонтекста* необходима, если требуется выполнение процедуры/функции и на сервере и на клиенте. Данная директива используется редко, но является полезной и позволяет вместо двух процедур, для выполнения в разных контекстах, указать всего одну. Последняя директива *&НаКлиентеНаСервере* по сути аналогична *&НаКлиентеНаСервереБезКонтекста*, но доступна она только в модуле команд.

Если по ошибке не указать директиву, то платформа сама ее поставит. По умолчанию это директива *&НаСервере*.

3 События формы

3.1 Обработчики событий элементов формы

Для определения поведения прикладных объектов отличного от стандартного и создания собственных алгоритмов их поведения используется встроенный язык. Программный код выполняется в заранее известных ситуациях, называемых событиями.

Такие процедуры, которые система будет вызывать сама, при наступлении какого-либо события, называются обработками событий. Найти их можно в форме документа выбираем элемент формы, открываем окно свойств - и в самом низу раздел "События". Изменить (добавить новые) состав событий, которые можно обработать нельзя. Можно только использовать существующие (рис. 38).

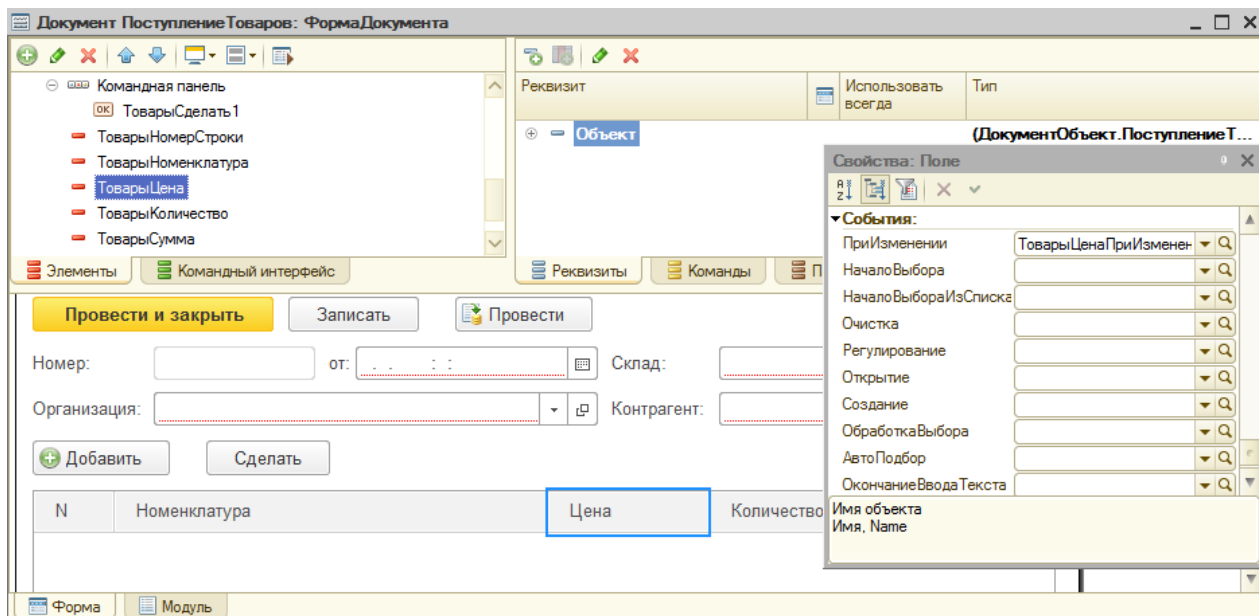


Рис.8 События элемента формы

Как правило, это событие ПриИзменении, которое возникает после изменения значения поля. Найдем его в списке событий и нажмем кнопку открытия со значком лупы в поле ввода. Система предложит окно для выбора типа обработчика события, который мы хотим создать (рис. 9).

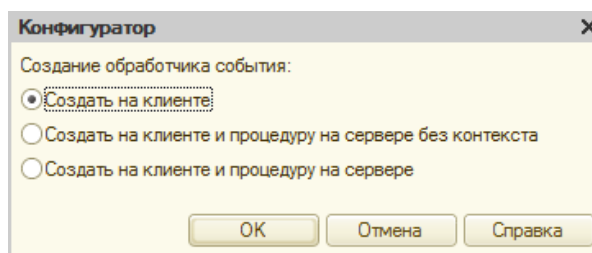


Рис. 9 Окно для выбора типа создаваемого обработчика события

После этого система создаст шаблон процедуры *обработчика* этого события в модуле нашей формы и откроет закладку Модуль редактора. *Модуль* – это «хранилище» для текста программы на встроенном языке. Модулей существует несколько, и каждый из них предназначен для описания алгоритмов, относящихся к тому или иному моменту работы программы. В данном случае это модуль формы, так как обработчики всех интерактивных событий, связанных с элементами формы, помещаются именно в модуль формы (рис.10).

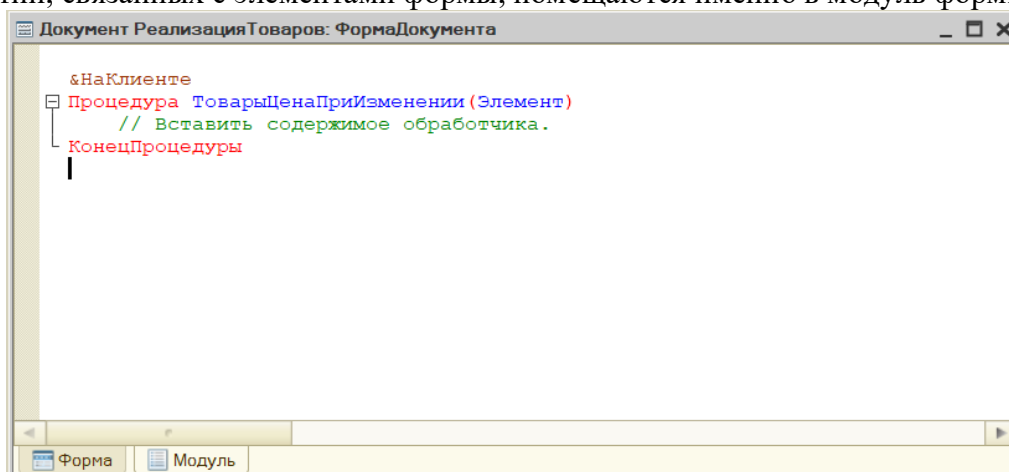


Рис. 10 Окно модуля формы документа

Когда создается форма, например, форма элемента документа, то платформа 1С автоматически создает на форме основной реквизит с именем **Объект**. Через данный реквизит доступны свойства текущего объекта, который редактируется на форме.

Например, можно получать ссылку на текущий открытый элемент справочника:

```
СсылкаНаТекущийЭлемент=Объект.Ссылка;
```

или получить значение реквизита в форме документа

```
Объект.Реквизит = "значение";
```

Работа с табличной частью отличается от работы с реквизитами, поскольку это встроенная таблица и содержит несколько значений для одного экземпляра объекта.

Рассмотрим некоторые приемы работы с табличными частями.

1) Как обойти табличную часть. Для обхода табличной части можно использовать цикл *Для каждого*

```
Для каждого Строка из ТабличнаяЧасть Цикл
```

```
Сообщить (Строка.РеквизитТабличнойЧасти);
```

```
КонецЦикла;
```

На каждой итерации в переменную *Строка* передается очередная строка табличной части. Значения реквизитов строки можно получить выражением *Строка.ИмяРеквизита*.

2) Как очистить табличную часть

```
ТабличнаяЧасть.Очистить();
```

3) Как получить текущую строку табличной части. Текущая строка — это строка, в которой у пользователя в данный момент находится курсор. Чтобы ее получить, нужно обратиться к элементу управления на форме, который связан с табличной частью.

```
Элементы.ИмяТабличногоПоля.ТекущиеДанные;
```

или всю строку полностью

```
Элементы.ТабличнаяЧасть.ТекущиеДанные;
```

5) Как добавить новую строку в табличную часть. Добавление новой строки в конец табличной части:

```
НоваяСтрока=ТабличнаяЧасть.Добавить();
```

```
НоваяСтрока.Реквизит1="Значение";
```

Добавление новой строки в любое место табличной части (последующие строки будут сдвинуты):

```
НоваяСтрока=ТабличнаяЧасть.Вставить(Индекс)
```

```
//Индекс – номер добавляемой строки. Нумерация строк начинается с нуля.
```

```
НоваяСтрока.Реквизит1="Значение";
```

6) Как манипулировать реквизитами табличной части

```
ВыделеннаяСтрока=Элементы.ТабличнаяЧасть.ТекущиеДанные;
```

```
ВыделеннаяСтрока.Реквизит1=ВыделеннаяСтрока.Реквизит2 + 10;
```

3.2 Экспортируемые процедуры и общие модули

Любая процедура, функция, переменная, объявленная в модуле, видна только в этом модуле. Если нам необходимо получить возможность вызова процедур (функций), такие процедуры (функции) должны быть экспортируемыми. Экспортируемые процедуры, функции, переменные отмечаются в тексте модуля ключевым словом «Экспорт».

Например, у нас существует необходимость внести расчеты суммы строки табличной части в отдельную процедуру, и вызывать ее из обработчика событий различных объектов.

Для это служат объекты конфигурации *Общий модуль*, расположенные в ветке **Общие / Общие модули**. Процедуры и функции, содержащиеся в этих модулях, могут быть доступны для любых объектов конфигурации.

Для создания общего модуля, раскроем ветвь **Общие** в дереве объектов конфигурации, нажав на + слева от нее. Затем выделим ветвь **Общие модули** и нажмем кнопку **Добавить** в командной панели окна конфигурации (рис. 11).

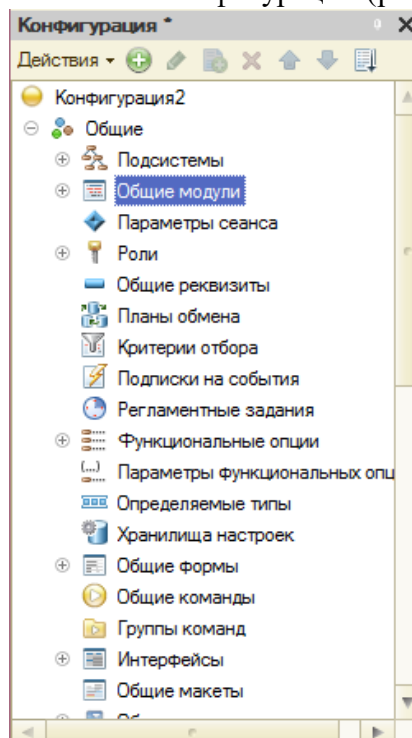


Рис. 11. Создание общего модуля в дереве объектов конфигурации

Общие модули представляют собой библиотеки процедур и функций: такой модуль не может содержать раздела объявления переменных и тела модуля, только процедуры и функции. Как правило, они экспортируемые, исключение составляют лишь те процедуры и функции модуля, которые используются только в этом же модуле.

Называют модули, как правило, так, чтобы было понятно, какие процедуры там содержатся и где исполняются (рис.12).

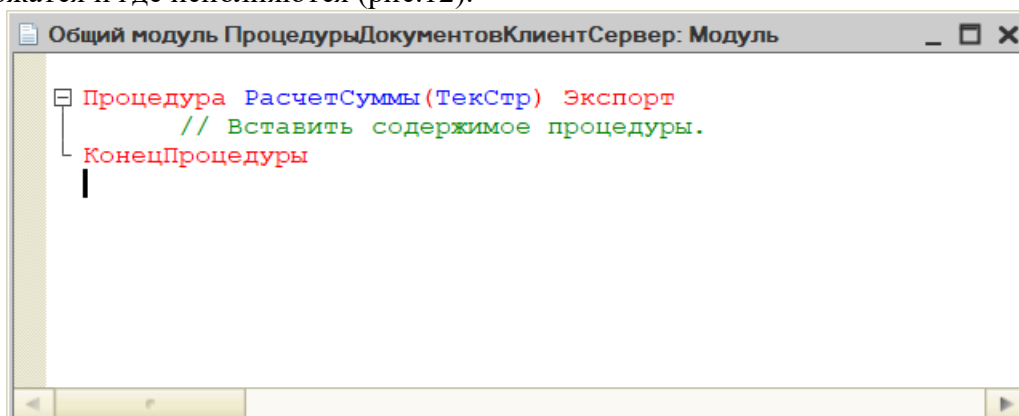


Рис.12 Общий модуль

Ключевое слово **Экспорт** в заголовке процедуры указывает на то, что эта процедура экспортируемая и может быть доступна из других программных модулей.

Обращение к экспортной процедуре осуществляется через имя модуля:

```
ИмяМодуля.ИмяПроцедуры(Параметр1, Параметр2);
```

3.3 Обращение к данным информационной базы

Часто появляются ситуации, когда необходимо получить реквизит от ссылочного значения, хранящегося в реквизите формы. Например, у документа **Накладная** есть реквизит ссылочного типа – **Поставщик**, необходимо получить **ИНН** этого поставщика из реквизита справочника.

Мы находимся в модуле формы на клиенте. В контексте клиента ссылочные типы очень сильно ограничены в своих возможностях. В частности, нельзя получить значение реквизита от ссылки «через точку». Это можно сделать только в контексте сервера. Поэтому для решения этой задачи в модуле формы создается серверная внеконтекстная функция, которую вызываем с клиента и передаем в нее ссылку на поставщика. В теле функции, на сервере, мы получаем **ИНН** поставщика и веозвращаем его на клиента.

Например, локальная команда формы, вызывающая серверную функцию, если ссылка на поставщика располагается в шапке документа будет выглядеть следующим образом:

```
&НаКлиенте
Процедура ИмяПроцедуры (Команда)
    ИННПоставщика = ИмяПроцедурыНаСервере (Объект.Поставщик) ;
КонецПроцедуры
```

Или если ссылка на поставщика располагается в табличной части документа будет выглядеть следующим образом:

```
&НаКлиенте
Процедура ИмяПроцедуры (Команда)
    Переменная1=Элементы.ТабличнаяЧасть.ТекущиеДанные;
    Переменная1.ИННПоставщика=ИмяПроцедурыНаСервере (Переменная1.Поставщик) ;
КонецПроцедуры
```

А функция **ИмяПроцедурыНаСервере()** описывается здесь же, в модуле формы:

```
&НаСервереБезКонтекста
функция ИмяПроцедурыНаСервере (СсылкаНаПоставщика)

    Возврат СсылкаНаПоставщика.ИНН;

КонецФункции
```

Таким образом на сервере мы сразу же получаем значение реквизита «через точку» от полученной ссылки и возвращаем его на клиента.

Важно заметить, что функция **ИмяПроцедурыНаСервере()** не использует контекст формы (директива компиляции **&НаСервереБезКонтекста**). Потому что все, что нужно передать на сервер, – это только ссылка, и мы передаем ее в параметре функции. Использовать контекстный вызов и передавать на сервер весь контекст формы только ради того, чтобы на сервере взять из него значение одного реквизита формы, - это слишком расточительно.

Задание к лабораторной работе

1. Для документа **Приходная накладная** создайте форму документа. И выполните следующие действия:

- для элемента табличной части формы **Количество** установите событие **ПриИзменении** вариант **Создать на клиенте** и опишите обработчик события, в котором автоматически рассчитывается поле **Сумма** в строке табличной части ($Сумма = Количество * Цена$). **Обратите внимание**, что это событие табличной части документа, поэтому первоначально нужно получить текущую строку табличной части.

- для элемента табличной части формы **Цена** установите событие **ПриИзменении** и опишите обработчик события, в котором автоматически рассчитывается поле **Сумма** в строке табличной части ($Сумма = Количество * Цена$). - для элемента табличной части формы **Сумма** установите событие **ПриИзменении** и опишите обработчик события, в котором автоматически рассчитывается поле **Цена** в строке табличной части ($Цена = Сумма / Количество$). Делить на ноль нельзя (вдруг пользователь не успел ввести количество), поэтому нужно это учесть и использовать условный оператор.

- проверьте правильность выполнения событий в пользовательском режиме.

2. Добавьте объект конфигурации - **Общий модуль** с именем **РаботаСДокументами** со свойством **Клиент** включено, а свойство - **Сервер** выключено. В тексте модуля опишите экспортную процедуру, в качестве параметра передается текущая строка табличной части. В процедуре опишите расчет суммы $Сумма = Количество * Цена$. Замените строку расчета суммы в обработчиках событий **КоличествоПриИзменении()** и **ЦенаПриИзменении()** на строку вызова процедуры **РаботаСДокументами**.

3. Для документа **Расходная накладная** создайте форму документа. И выполните следующие действия (**обратите внимание**, что порядок написания обработчиков событий обратный выполнению действий пользователем):

- для элемента табличной части формы **СуммаНДС** установите событие **ПриИзменении** и опишите обработчик события, в котором рассчитывается поле **Всего**, как сумма по полю **СуммаНДС** и **Сумма**;

- для элемента табличной части формы **СтавкаНДС** установите событие **ПриИзменении** и опишите обработчик события, в котором поле **СуммаНДС** автоматически рассчитывается. Для этого необходимо запросить **Ставку** из справочника **СтавкиНДС** и умножить ее на значение в поле **Сумма** (обратите внимание, что ставка указывается в процентах, поэтому необходимо $Ставку / 100$, прежде, чем умножить на **Сумму**). Далее дополните обработчик события пересчетом поля **Всего** (можно с помощью вызова процедуры события поля **СуммаНДС**);

- для элемента табличной части формы **Сумма** установите событие **ПриИзменении** и опишите обработчик события, такой же как для элемента **СтавкаНДС** (можно с помощью вызова процедуры события поля **СтавкаНДС**);

- для элемента табличной части формы **Цена** установите событие **ПриИзменении** и опишите обработчик события, в котором автоматически рассчитывается поле **Сумма** в строке табличной части. Причем сделайте это через вызов экспортной процедуры общего модуля. Дополните обработчик события пересчетом поля **Сумма** (можно с помощью вызова процедуры события поля **Сумма**);

- для элемента табличной части формы **Количество** установите событие **ПриИзменении** и опишите обработчик события, такой же как для элемента **Цена** (можно с помощью вызова процедуры события поля **Цена**);

- для элемента табличной части формы **Номенклатура** установите событие **ПриИзменении** и опишите обработчик события, в котором поле **СтавкаНДС** автоматически заполняется данными из справочника **Номенклатура**. Дополните обработчик события пересчетом поля **СтавкаНДС** (можно с помощью вызова процедуры события поля **СтавкаНДС**). проверьте правильность выполнения событий в пользовательском режиме.

4. Выполните индивидуальное задание согласно варианту.

Варианты заданий к лабораторной работе

Вариант 1 - Фирма по оказанию мелкобытовых услуг

- 1) Определите обработчик события, который позволил бы после выбора сотрудника автоматически устанавливать в документе значение должности.
- 2) Добавьте реквизит **СрокПоставки** в справочник Номенклатура. Дополните табличную часть документа **ЗаказЗапЧастей** реквизитом **СрокПоставки** и определите обработчик события табличной части документа **ЗаказЗапЧастей**, которые позволили бы после выбора номенклатуры автоматически устанавливать в документе значение срока поставки.

Вариант 2 - Оптовая база детских игрушек

- 1) Определите обработчик события, который позволил бы после выбора склада автоматически устанавливать в документах **Приходная накладная** и **Расходная накладная** значение кладовщика.
- 2) Добавьте реквизит **Вес** в справочник Номенклатура. Дополните табличную часть документа **ЗаявкаНаДоставку** реквизитом **ВесВсего** и определите обработчик события табличной части документа **ЗаявкаНаДоставку**, которые позволили бы после выбора номенклатуры автоматически устанавливать в документе значение веса всей номенклатуры ($\text{Вес} * \text{Количество}$).

Вариант 3 - Ателье по ремонту и пошиву одежды

- 1) Определите обработчик события, который позволил бы после выбора контрагента автоматически устанавливать в документе **Расходная накладная** значение скидки.
- 2) Добавьте в табличную часть документа **Расходная накладная** реквизит **ВсегоСоСкидкой** и определите обработчик события, который позволил бы после изменения цены или количества автоматически рассчитывать значение всего с учетом скидки контрагента.

Вариант 4 - Фирма по ремонту телефонов

- 1) Определите обработчик события, который позволил бы после выбора контрагента автоматически устанавливать в документе **Расходная накладная** значение города.
- 2) Добавьте реквизит **Комплектность** в справочник Номенклатура. Дополните табличную часть документа **ЗаданиеНаРаботу** реквизитом **Комплектность** и определите обработчик события табличной части документа **ЗаданиеНаРаботу**, которые позволили бы после выбора номенклатуры автоматически устанавливать в документе значение комплектности.

Вариант 5 - Фирма по ремонту автомобилей

- 1) Определите обработчик события, который позволил бы после выбора контактного лица автоматически устанавливать в документе **Расходная накладная** значение контрагента. Предварительно нужно сбросить у контактных лиц настройку свойства «Связь параметров выбора».
- 2) Добавьте реквизит **НормаВремени** в справочник Номенклатура. Дополните табличную часть документа **Заказ-наряд** реквизитом **ВремяРаботы** и определите обработчик события табличной части документа **Заказ-наряд**, которые позволили бы после выбора номенклатуры автоматически устанавливать в документе значение времени работы ($\text{НормаВремени} * \text{Количество}$).

Вариант 6 - Химчистка

1) Определите обработчик события, который позволил бы после выбора номенклатуры устанавливать в документе **Расходная накладная** значение валюты.

2) Добавьте реквизит **Банк** в справочник Контрагенты. Дополните документ **ОплатаПоставщику** реквизитом **БанкПолучателя** и определите обработчик события, которые позволили бы после выбора контрагента автоматически устанавливать в документе значение банка получателя.

Вариант 7 – Типография

1) Определите обработчик события, который позволил бы после выбора номенклатуры устанавливать в документе **Расходная накладная** значение количества упаковок, как разность по полю Количество/КоличествоВУпаковке.

2) Добавьте реквизит **Должность** в справочник Сотрудники. Дополните документ **ПланированиеРесурсов** реквизитами **Сотрудник** и **Должность** и определите обработчик события, которые позволили бы после выбора сотрудника автоматически устанавливать в документе значение его должности.

Вариант 8 - Салон красоты

1) Определите обработчик события, который позволил бы после выбора номенклатуры устанавливать в документе **Расходная накладная** значение единицы измерения.

2) Добавьте реквизит **ТипМастера** в справочник Сотрудники. Дополните документ **ЗаписьКМастеру** реквизитом **ТипМастера** и определите обработчик события, которые позволили бы после выбора сотрудника автоматически устанавливать в документе значение типа мастера.

Вариант 9 - Медицинский центр

1) Определите обработчик события, который позволил бы после выбора номенклатуры устанавливать в документе **Расходная накладная** значение серии.

2) Добавьте реквизит **ТипСпециалиста** в справочник Сотрудники. Дополните документ **РасписаниеВрача** реквизитом **ТипСпециалиста** и определите обработчик события, которые позволили бы после выбора сотрудника автоматически устанавливать в документе значение типа специалиста.