|  |
| --- |
| титул |
| **115172, Москва, ул. Б.Каменщики, д. 7;тел., факс: (495) 134 1234; e-mail: spo-54@edu.mos.ru** |

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ**

к лабораторной работе №8:

**Обработчик событий. Структура модуля. Расчет суммы в документе.**

по учебной дисциплине

**МДК 01.01 Разработка программных модулей**

Специальность

**09.02.07 «Информационные системы и программирование»**

**Оглавление**

[**Введение** 3](#_Toc148363013)

[**1.** **Программирование — обработчики событий** 11](#_Toc148363014)

[**2.** **Программирование - области (структура модуля)** 17](#_Toc148363015)

[**Индивидуальное задание** 21](#_Toc148363016)

[**Контрольные вопросы** 22](#_Toc148363017)

# **Введение**

**Цель работы**: Знакомство с обработчиком событий. Изучение структуры модулей в 1С:Предприятие. Добавление автоматического расчета суммы в документах..

**Время выполнения**: 2 часа

**Задание**:

1. Законспектировать теоретические сведения, при наличии.

2. Выполнить лабораторную работу.

3. Выполнить индивидуальное задания, при наличии.

4. Защитить работу, ответив на контрольные вопросы устно.

Особенностью обработки событий среде 1С:Предприятия 8 является то, что имя процедуры-обработчика в одних случаях должно совпадать с именем события, а в других случаях может от него отличаться.

Особенности структуры модуля

1.1. В программном модуле (общие модули, модули объектов, модули менеджеров объектов, модули форм, команд и т.п.) в общем случае могут присутствовать следующие разделы в приведенной ниже последовательности:

* заголовок модуля;
* раздел описания переменных;
* экспортные процедуры и функции модуля, составляющие его программный интерфейс;
* обработчики событий объекта (формы);
* служебные процедуры и функции модуля;
* раздел инициализации.

1.2. Объемные разделы модулей рекомендуется разбивать на подразделы по функциональному признаку.

1.3. Разделы и подразделы оформляются в виде областей. При этом имена областей должны удовлетворять требованиям стандарта Правила образования имен переменных

1.4. Шаблон разделов для общих модулей:

* Раздел «Программный интерфейс» содержит экспортные процедуры и функции, предназначенные для использования другими объектами конфигурации или другими программами (например, через внешнее соединение);

#Область ПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

* Раздел «Служебный программный интерфейс» предназначен для модулей, которые являются частью некоторой функциональной подсистемы. В нем должны быть размещены экспортные процедуры и функции, которые допустимо вызывать только из других функциональных подсистем этой же библиотеки;

#Область СлужебныйПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

* Раздел «Служебные процедуры и функции» содержит процедуры и функции, составляющие внутреннюю реализацию общего модуля. В тех случаях, когда общий модуль является частью некоторой функциональной подсистемы, включающей в себя несколько объектов метаданных, в этом разделе также могут быть размещены служебные экспортные процедуры и функции, предназначенные только для вызова из других объектов данной подсистемы.

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

1.5. Шаблон оформления разделов для модулей объектов, менеджеров, наборов записей, обработок, отчетов и т.п.:

* Раздел «Программный интерфейс» содержит экспортные процедуры и функции, предназначенные для использования в других модулях конфигурации или другими программами (например, через внешнее соединение). Не следует в этот раздел помещать экспортные функции и процедуры, которые предназначены для вызова исключительно из модулей самого объекта, его форм и команд.

#Область ПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

* Раздел «Обработчики событий» содержит обработчики событий модуля объекта (ПриЗаписи, ПриПроведении и др.);

#Область ОбработчикиСобытий

// Код процедур и функций

#КонецОбласти

* Раздел «Служебный программный интерфейс» имеет такое же предназначение, как и в общих модулях;

#Область СлужебныйПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

* Раздел «Служебные процедуры и функции» имеет такое же предназначение, как и в общих модулях.

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

1.6. Шаблон оформления разделов для модулей форм:

* Раздел «Обработчики событий формы» содержит процедуры-обработчики событий формы: ПриСозданииНаСервере, ПриОткрытии и т.п.;

#Область ОбработчикиСобытийФормы

// Код процедур и функций

#КонецОбласти

* Раздел «Обработчики событий элементов шапки формы» содержит процедуры-обработчики элементов, расположенных в основной части формы (все, что не связано с таблицами на форме);

#Область ОбработчикиСобытийЭлементовШапкиФормы

// Код процедур и функций

#КонецОбласти

* В разделах «Обработчики событий элементов таблицы формы <имя таблицы формы>» размещаются процедуры-обработчики таблиц формы и элементов таблиц. Для процедур-обработчиков каждой таблицы должен быть создан свой раздел;
* #Область ОбработчикиСобытийЭлементовТаблицыФормы <ИмяТаблицыФормы>

// Код процедур и функций

#КонецОбласти

* Раздел «Обработчики команд формы» содержит процедуры-обработчики команд формы (имена которых задаются в свойстве Действие команд формы);

#Область ОбработчикиКомандФормы

// Код процедур и функций

#КонецОбласти

* Раздел «Служебные процедуры и функции» имеет такое же предназначение, что и в общих модулях.

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

1.7. Шаблон оформления разделов для модулей команд:

* Раздел «Обработчики событий» содержит процедуру-обработчик команды ОбработкаКоманды;

#Область ОбработчикиСобытий

// Код процедур и функций

#КонецОбласти

* Раздел «Служебные процедуры и функции» имеет такое же предназначение, что и в общих модулях.

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

1.8. В модуле не должно быть пустых областей.

2. Общие требования к разделам программных модулей.

2.1. Заголовок модуля представляет собой комментарий в самом начале модуля. В заголовке модуля приводится его краткое описание и условия применения.

Например:

////////////////////////////////////////////////////////////////////////////////

// Клиентские процедуры и функции общего назначения:

// - для работы со списками в формах;

// - для работы с журналом регистрации;

// - для обработки действий пользователя в процессе редактирования

// многострочного текста, например комментария в документах;

// - прочее.

////////////////////////////////////////////////////////////////////////////////

Для модулей форм в заголовке рекомендуется размещать описание параметров формы.

2.2. Раздел описания переменных. Имена переменных назначаются согласно общим правилам образования имен переменных, а их использование описывается в статье Использование глобальных переменных в программных модулях.

Все переменные модуля должны быть снабжены комментарием, достаточным для понимания их назначения. Комментарий рекомендуется размещать в той же строке, где объявляется переменная.

Пример:

#Область ОписаниеПеременных

Перем ВалютаУчета;

Перем АдресПоддержки;

...

#КонецОбласти

2.3. Программный интерфейс. Экспортные процедуры и функции, составляющие программный интерфейс модуля, размещаются сразу же после описания переменных. Такие процедуры и функции предназначены для использования другими объектами конфигурации или другими программами (например, через внешнее соединение), поэтому должны быть расположены в модуле на "видном месте".

2.4. У каждого события должна быть назначена своя процедура-обработчик. Если одинаковые действия должны выполняться при возникновении событий в разных элементах формы следует:

* создать отдельную процедуру (функцию), выполняющую необходимые действия;
* для каждого элемента формы создать отдельный обработчик с именем, назначаемым по умолчанию;
* из каждого обработчика вызвать требуемую процедуру (функцию).

2.5. Обработчики событий модулей объекта и менеджера объекта размещаются после раздела с программным интерфейсом, но до служебных процедур и функций модуля.

2.6. Служебные процедуры и функции модуля, которые не являются обработчиками событий, а составляют внутреннюю реализацию модуля, размещаются в модуле следом за обработчиками событий.

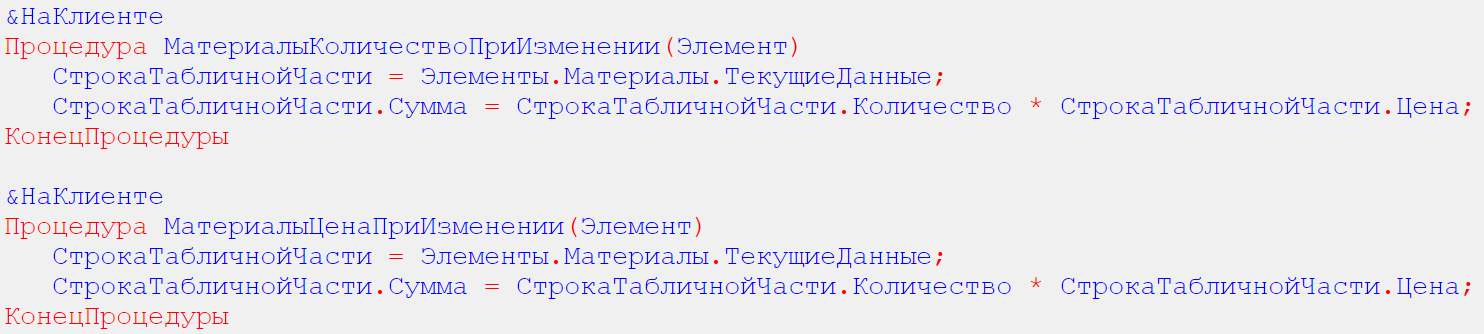
2.7. Раздел инициализации содержит операторы, инициализирующие переменные модуля или объект (форму).

Расчет суммы в строке табличной части справочника или документа.

В реальной работе прикладного решения часто возникает необходимость автоматизировать некоторые подсчеты согласно собственному алгоритму. Например, сделать так, чтобы сумма в строке табличной части документа автоматически вычислялась при изменении цены или количества в строке.

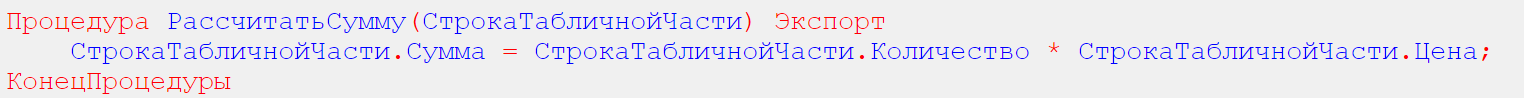
Это небольшое вычисление производится в модуле формы в процедуре, выполняющейся на клиенте. Такой процедурой являются обработчики события ПриИзменении полей Цена и Количество табличной части документа.

Итак, сначала нужно создать форму документа, чтобы в ней описать собственный алгоритм вычислений. Затем, открыв палитру свойств полей Цена и Количество табличной части документа, создать для них обработчики события ПриИзменении и внести в них следующий код:

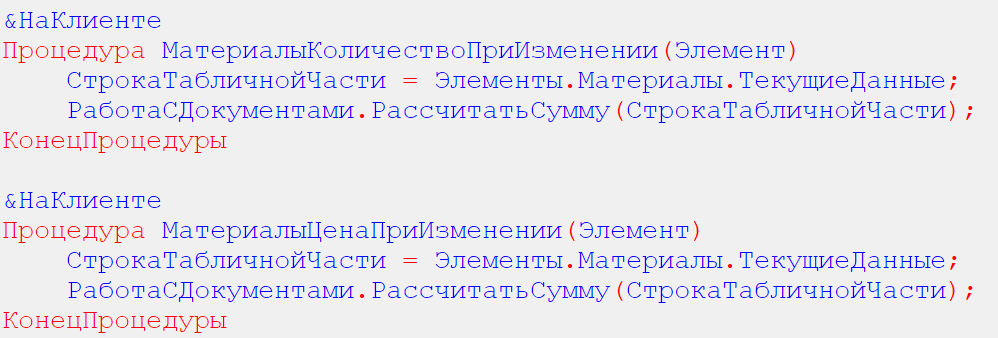


При выполнении этого кода в переменную СтрокаТабличнойЧасти помещается объект, содержащий данные текущей строки документа, которую требуется пересчитать. Получив этот объект, можно обратиться к данным конкретной колонки табличной части, указав имя колонки в качестве свойства объекта. Таким образом, во второй строке процедуры обработчика вычисляется значение колонки Сумма как произведение значений колонок Количество и Цена.

Если процедура расчета суммы может понадобиться сразу для нескольких документов, то полезно разместить эту процедуру в общем модуле, и обращаться к ней из форм документов. Например, в общем модуле РаботаСДокументами поместить процедуру РассчитатьСумму():



В модуле формы документа вызывать ее из обработчиков события **ПриИзменении**:



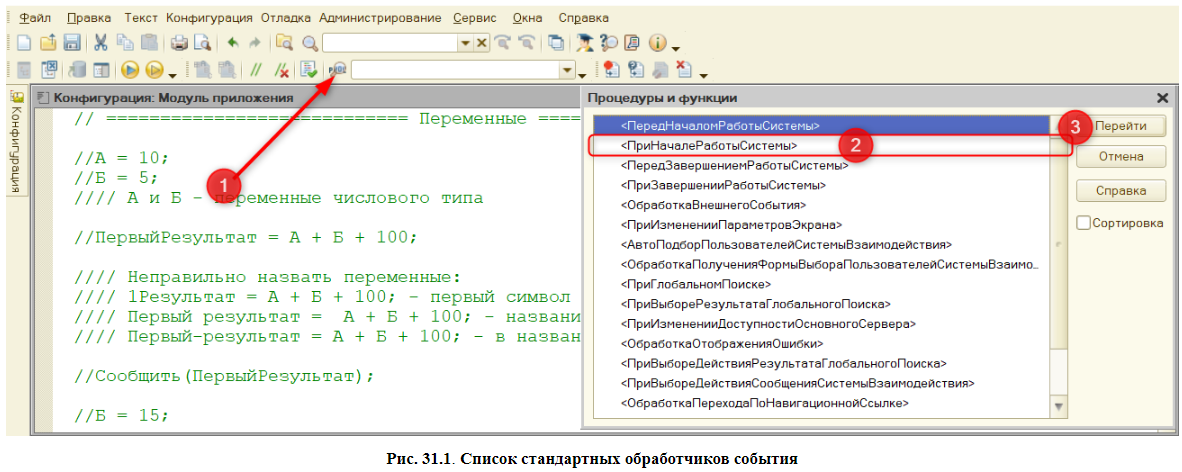
# **Программирование — обработчики событий**

Познакомимся с важной составляющей, которая является ключевым элементом при разработке прикладных решений — это обработчики событий.

Обработчик события — метод (процедура или функция), который исполняется в момент наступления какого-либо события в программе. Примеры событий: запуск программы, закрытие окна, запись элемента справочника, изменение поля и др.

Для начала посмотрим простой пример обработки запуска программы. Чтобы посмотреть весь список обработчиков события в модуле приложения, нужно нажатьна кнопку «Процелуры и функции». Также эта кнопка находится и в пункте главного меню «Текст >Процедуры и функции».

После нажатия на кнопку открывается окно со списком стандартных обработчиков событий, которые есть в данном модуле. Их достаточно много, но на данный момент нам нужен обработчик «ПриНачалеРаботыСистемы» (рис. 31.1).

****

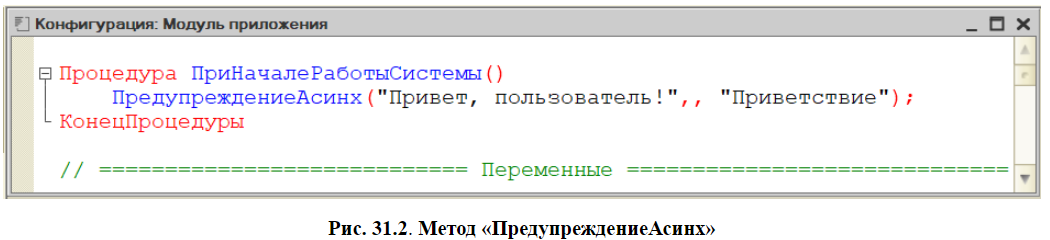
**Рисунок 31.1 Список стандартных обработчиков события**

Обработчик «ПриНачалеработыСистемы» срабатывает после того, как программа запустилась.

После выбора в модуле прописывается тело процедуры. Если изменить название процедуры, то алгоритм исполняться не будет: срабатывание процедуры привязано к названию.

Напишем в данной процедуре простое приветствие. Воспользуемся для этого другим методом, который позволит вывести отдельное окно пользователю.

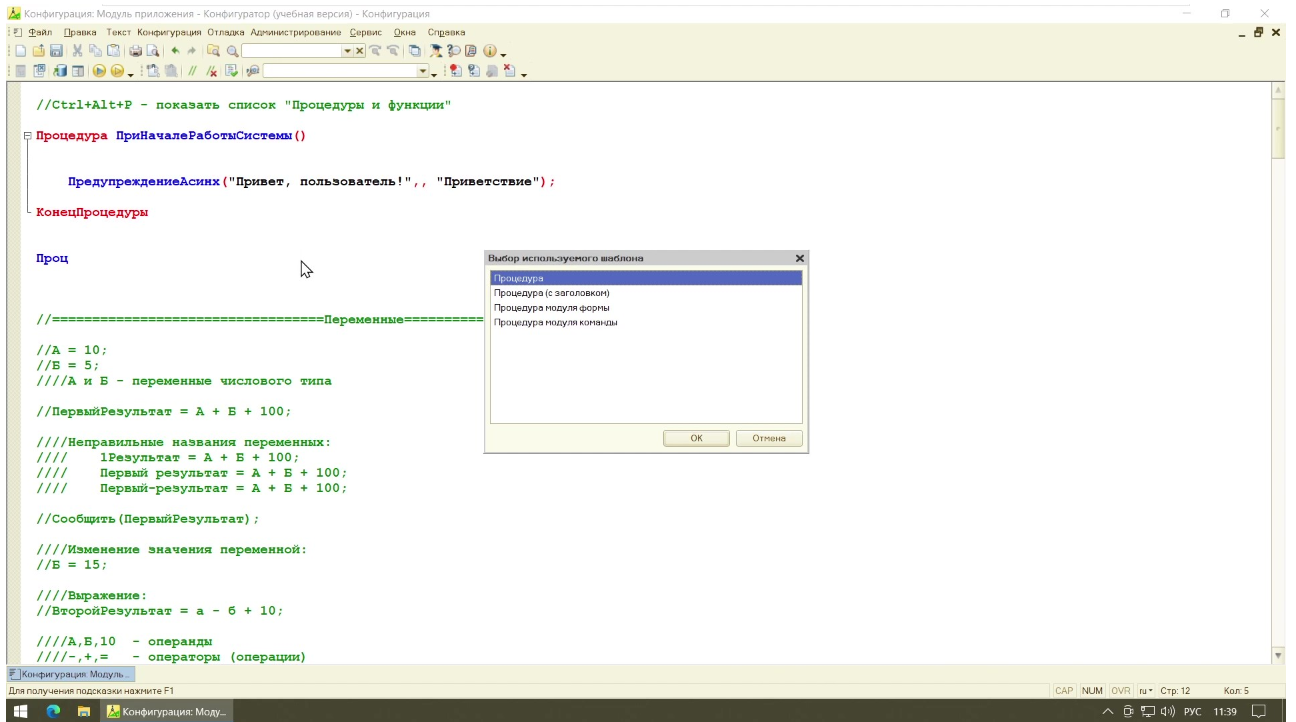
Пропишем алгоритм, используя метод «Предупреждениедсинх» (рис. 31.2):

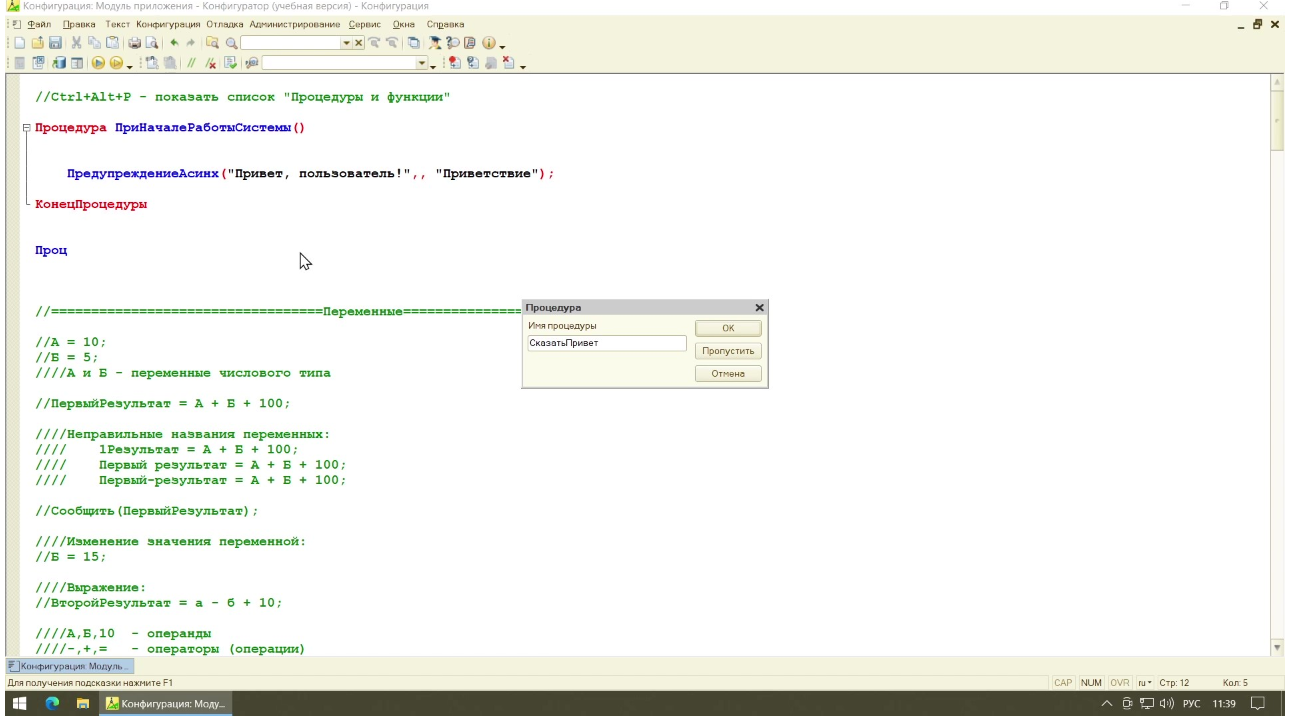
****

**Рисунок 31.2 Метод «ПредупреждениеАсинх»**

Далее опишем собственную процедуру, которая добавит разнообразия нашей задаче: пользователь будет сам вводить свое имя, а затем будет выводиться приветствие.

Чтобы не писать шаблон процедуры вручную. напишем четыре первых буквы «Проц» и нажмем на комбинацию «Ctrl+Q». Откроется окно со списком шаблонов в котором мы выберем первый пункт и нажмем «ОК». В следующем окне введем имя процедуры и нажмем «ОК» (рис. 31.3).

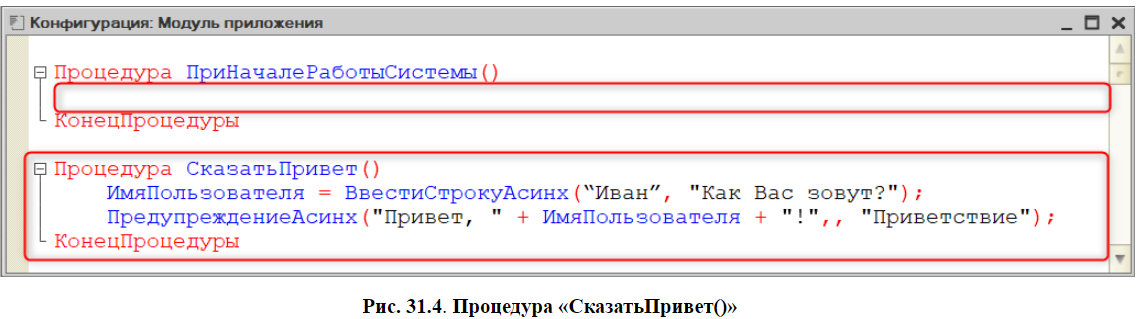


****

**Рис. 31.3. Создание шаблона процедуры**

Далее внутри процедуры опишем алгоритм, который будет спрашивать пользователя его имя и здороваться с ним, обращаясь по этому имени.

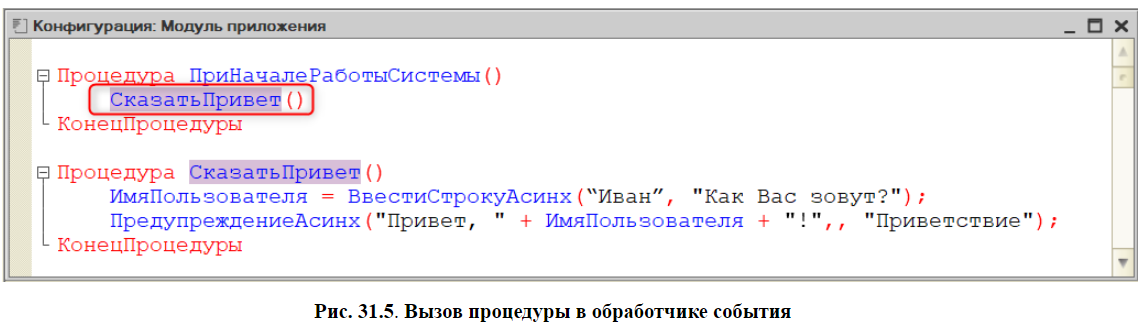
Для начала добавим переменную «ИмяПользователя» и присвоим ей значение. Имя будет вводиться в виде текста, что означает тип «Строка» для системы. Чтобы имя вводилось пользователем вручную. воспользуемся методом «ВвестиСтрокуАсинх». Далее напишем само приветствие с учетом имени пользователя (рис. 31.4):

****

**Рисунок 31.4 Процедура «СказатьПривет()»**

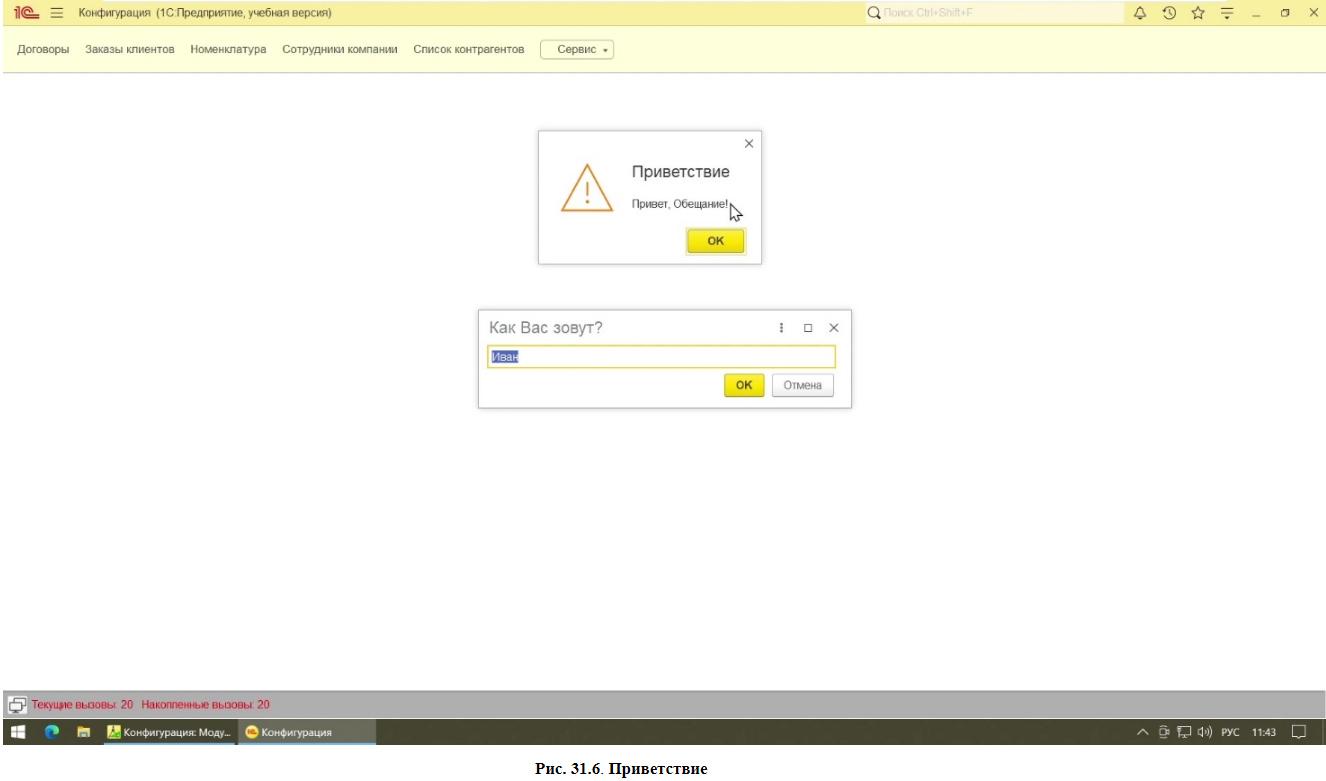
Обновим конфигурацию и проверим работу в пользовательском режиме. После запуска никаких окон предупреждения не открылось: это связано с тем, что в обработчике событий нет вызова нашей процедуры.

Чтобы вызвать процедуру, нужно указать ее имя (рис. 31.5).

****

**Рисунок 31.5 Вызов процедуры в обработчике события**

Еще раз обновим конфигурацию и запустим пользовательский режим. После запуска появятся сразу два окна: окно с приветствием и окно ввода имени. Получилось не так, как мы планировали (рис. 31.6).

****

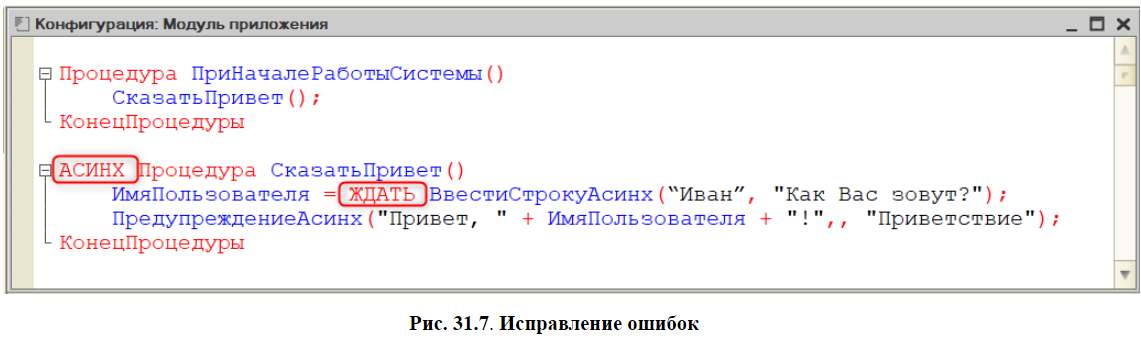
**Рисунок 31.6 Приветствие**

В выводе отобразилось «Обещание». Это связано с тем, что наша программа немного поторопилась: процедуры сработали друг за другом сразу. Нужно сделать так, чтобы вывод приветствия срабатывал только после того, как будет введено имя пользователя с клавиатуры.

Для этого перед вызовом метода «ВвестиСтрокуАсинх» воспользуемся служебным словом «Ждать».

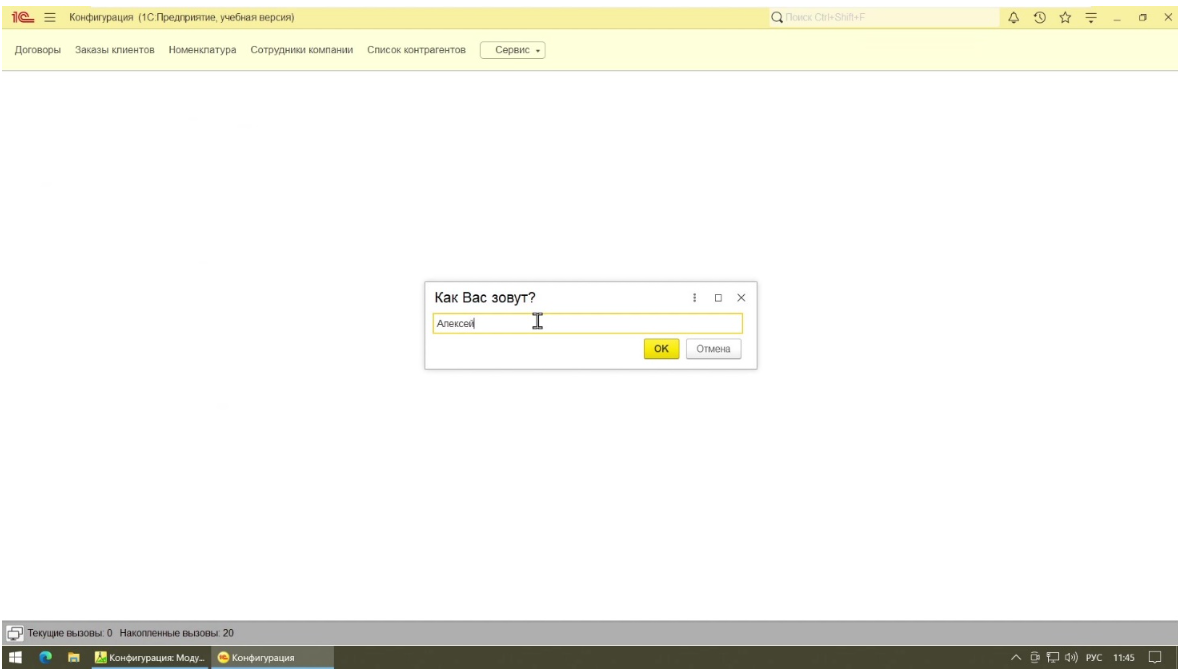
Если проверить модуль на синтаксические ошибки, то появится ошибка «Оператор «Ждать» может употребляться только в асинхронных процедурах или функциях».

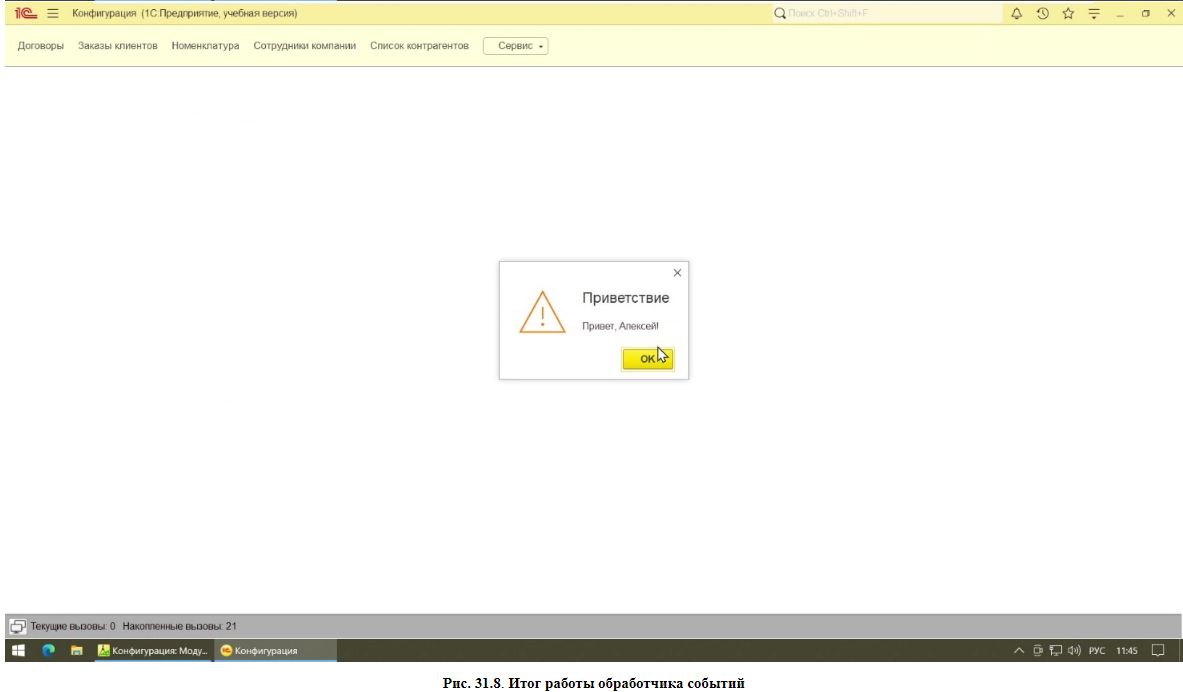
Чтобы исправить эту ошибку, требуется при определении процедуры указать, что она будет а синхронной (рис. 31.7).

****

**Рисунок 31.7 Исправление ошибок**

Обновим конфигурацию и запустим пользовательский режим. Проверим работу введя имя. Теперь «ПредупреждениеАсинх» работает корректно (рис. 31.8).



****

**Рисунок 31.8 Итог работы обработчика событий**

Итак, мы познакомились с обработчиками событий на базовом уровне.

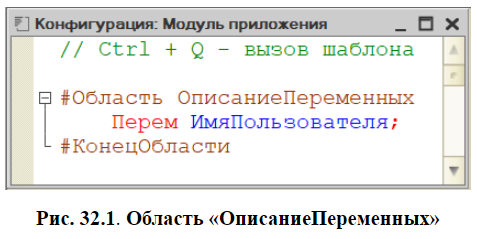
# **Программирование - области (структура модуля)**

В стандартах по разработке на языке 1С рекомендуется структурировать программный кол.

В структуре программного кода есть механизм, который позволяет упростить чтение модуля. Такой механизм называется «Область». Названия областей можно указывать произвольно. однако по стандартам есть фиксированные названия. Для удобства чтения и анализа кода рекомендуется использовать фиксированные названия. Например:

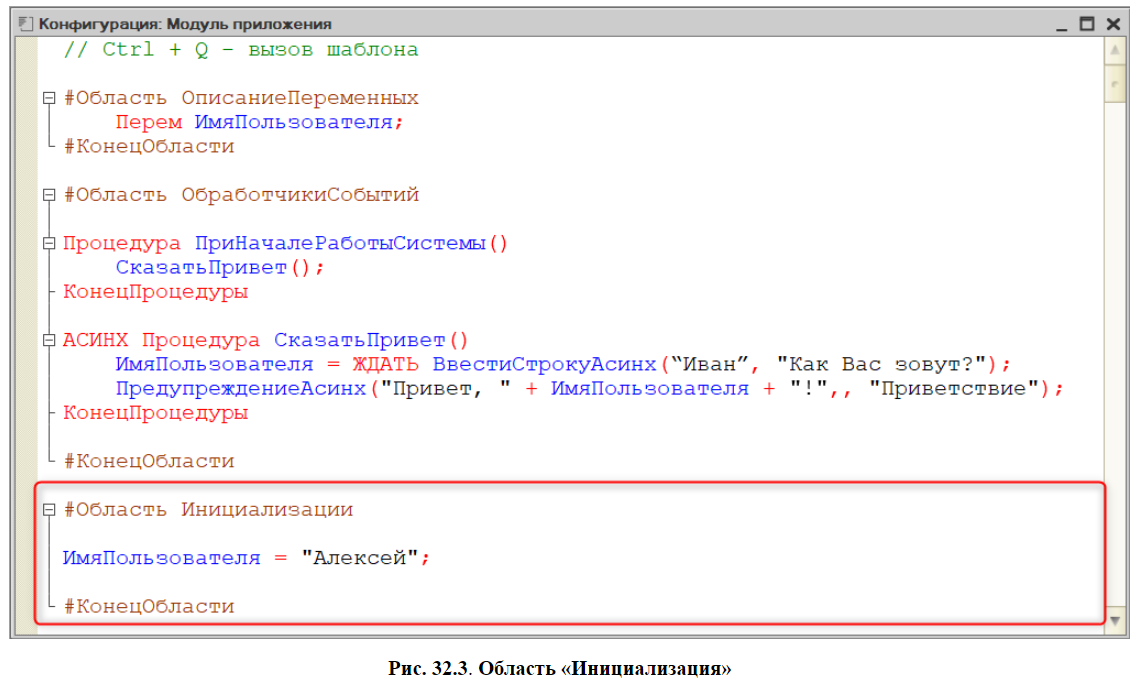
* **ОписаниеПеременных** — содержит описание глобальных, в рамках всего модуля, переменных.
* **ОбработчикиСобытий** — содержит обработчики событий модуля объекта.
* **Инициализация** — содержит операторы. инициализирующие переменные модуля (присваивание значений переменным).

Приведенные выше примеры областей опишем в нашем модуле приложения. Первую область «ОписаниеПеременных» запишем, используя символ «#», и конструкцию «Обл» — с помощью комбинании клавиш «CTRL-+Q». Чтобы определить переменную, используем служебное слово «Перем» (рис. 32.1).

****

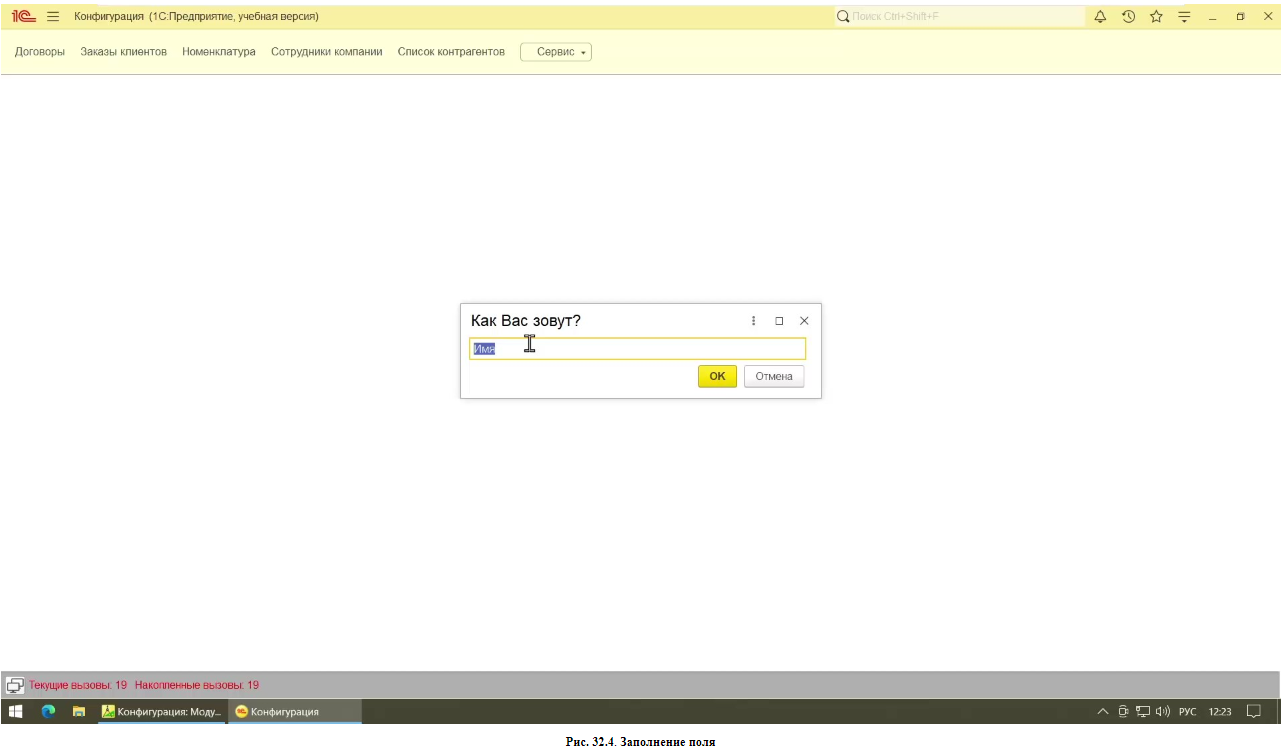
**Рисунок 32.1 Область «ОписаниеПеременных»**

Следующая описываемая область — это «Инициализация». В этой области запишем присваивание переменной «ИмяПользователя» значения «Имя» (рис. 32.3). Таким образом, пользователю при вводе имени изначально будет высвечиваться «Имя».

****

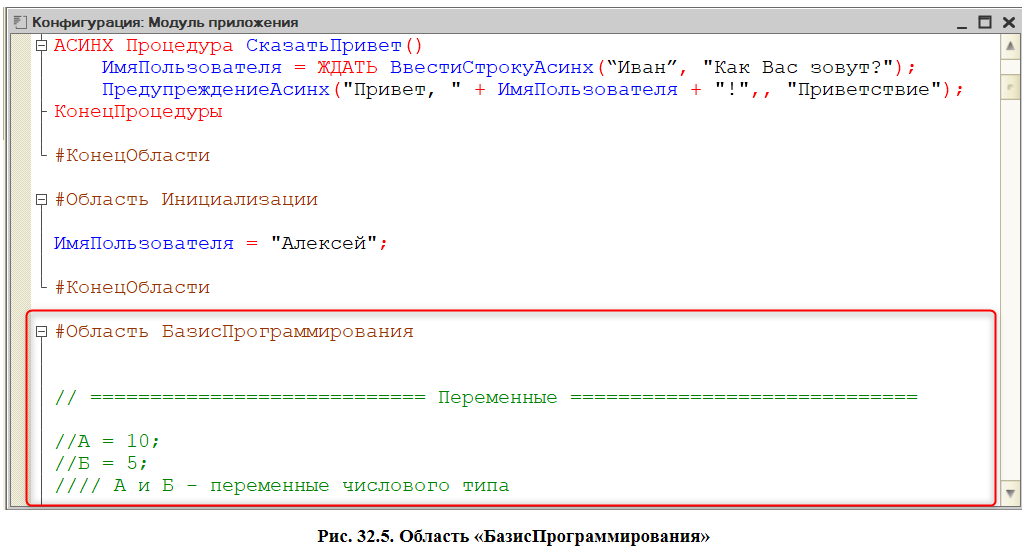
**Рисунок 32.3 Область «Инициализация»**

Обновляем конфигурацию и запускаем пользовательский режим. После запуска появляется окно ввода имени, изначально заполненное значением «Имя» (рис. 32.4).

****

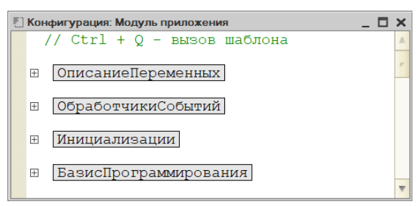
**Рисунок 32.4 Заполнение поля**

Теперь создадим область с произвольным названием. Назовем область «БазисПрограммирования» и поместим в нее все закомментированные ранее строчки в модуле (рис. 32.5).

****

**Рисунок 32.5 Область «БазисПрограммирования»**

Благодаря областям появилась возможность сворачивать и разворачивать группы кода, а также чтение и анализ модуля стали удобнее (рис. 32.6)

****

**Рисунок. 32.6 Сворачивание областей**

С точки зрения стандартов разработки, правильно разделять модуль на области. В таком одуле, поделенном на области, разработчику, не знакомому с нашим программным кодом, будет проще сориентироваться и разобраться.

# **Индивидуальное задание**

**№15 Алгоритмы – «Как тебя зовут?»**

При запуске программы необходимо спрашивать у человека его имя и дату рождения.

Если сегодняшний день – указанный день рождения, тогда мы должны поздравить именинника.

Если день рождения будет в ближайшие 5 дней, то пожелать пользователю хорошо провести праздник.

Если день рождения уже прошел, но еще не прошло5 дней после него, тогда мы должны спросить, как прошел праздник.

В ином случае – просто приветствовать пользователя.

Примечание: при выполнении задания необходимо использовать асинхронный метод ВвестиДатуАсинх() и ключевое слово Ждать.

# **Контрольные вопросы**

1. Что такое «Обработчик события»?
2. За что отвечает свойство «ТолькоПросмотр»?
3. Будет ли работать обработчик события, если его не привязать к событию поля и не вызвать из другого срабатывающего обработчика? Объясните почему?