

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Севастопольский государственный университет»**

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

к выполнению лабораторных работ  
по дисциплине «Платформа 1С»  
для студентов очной и заочной форм обучения  
по направлениям  
09.03.02 «Информационные системы и технологии»  
09.03.03 «Прикладная информатика»

Севастополь  
2021

УДК

Методические указания для выполнения лабораторных работ по дисциплине «Платформа 1С» для студентов направлений направлениям 09.03.02 «Информационные системы и технологии», 09.03.03 «Прикладная информатика» очной и заочной форм обучения / Сост.: (Вписать составителей) – Севастополь: СевГУ, 2021. – (Количество страниц) с.

Целью методических указаний является оказание помощи студентам в выполнении лабораторных работ по дисциплине. Приводятся постановка задачи для выполнения работы, варианты заданий, описаны этапы выполнения работы с приведенными примерами.

Методические указания рассмотрены и утверждены на заседании базовой кафедры (кафедру добавить) (протокол № \_\_ от \_\_ сентября 2021 г)

Рецензент:

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
1. ОБЩЕЕ ОПИСАНИЕ ВСТРОЕННОГО ЯЗЫКА.....	5
2. КОММЕНТАРИИ.....	5
3. ОПЕРАТОРЫ ВСТРОЕННОГО ЯЗЫКА .....	6
4. ВИДЫ И СТРУКТУРА ПРОГРАММНЫХ МОДУЛЕЙ .....	6
5. СОБЫТИЙНАЯ ОРИЕНТИРОВАННОСТЬ ВСТРОЕННОГО ЯЗЫКА.....	10
6. ПРИМИТИВНЫЕ ТИПЫ ДАННЫХ .....	11
7. СЛОЖНЫЕ ТИПЫ ДАННЫХ. УНИВЕРСАЛЬНЫЕ КОЛЛЕКЦИИ ЗНАЧЕНИЙ .....	18
8. КОНСТРУКЦИИ ЯЗЫКА .....	26
9. ПРИКЛАДНЫЕ ОБЪЕКТЫ «ОБРАБОТКИ».....	28
10. ОТЛАДЧИК .....	40
11. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС .....	42
11.1 Создание Подсистем .....	42
11.2 Интерфейс пользователя 1С. Основные сведения.....	44
11.3 Интерфейс «Такси» .....	46
11.4 Разработка пользовательского интерфейса.....	52
ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ .....	61
СОДЕРЖАНИЕ ОТЧЕТА.....	63
КОНТРОЛЬНЫЕ ВОПРОСЫ.....	63

## ВВЕДЕНИЕ

Технологическая платформа «1С:Предприятие» обеспечивает поддержку работы прикладных решений с различными операционными системами и СУБД, в том числе в среде открытого программного обеспечения и на мобильных устройствах. Открытость решений, возможность их динамичного развития, высокая функциональность и гибкость, возможность применения программ единой системы как в небольших предприятиях, так и в крупных организациях федерального масштаба обеспечивают высокую популярность решений на платформе «1С:Предприятия»: их использование широко распространено в России, Казахстане, Беларуси, они успешно применяются организациями многих других стран. Это создаёт большой рынок для программистов, владеющих языком и объектной моделью «1С:Предприятия».

Платформа «1С:Предприятие» является не универсальным, а специализированным, предметно-ориентированным средством разработки, предназначенным на решение задач автоматизации бизнеса. Одно из основных преимуществ этой предметно-ориентированной среды разработки – построение системы на основе технологической модели работы приложения, метаданных и прикладной модели работы приложения, что позволяет существенно упростить и ускорить разработку.

## ЛАБОРАТОРНАЯ РАБОТА №2. ВСТРОЕННЫЙ ЯЗЫК. СРЕДА РАЗРАБОТКИ И ОТЛАДКИ. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС.

Цель работы: ознакомиться с основными конструкциями встроенного языка 1С и методом построения пользовательских интерфейсов системы «1С:Предприятие. Получить базовые знания обработки пользовательских действий.

Используемое программное обеспечение: учебная версия платформы «1С:Предприятие».

### Теоретические сведения

#### 1. Общее описание встроенного языка

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

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

Вот лишь некоторые, наиболее значимые особенности встроенного языка:

- предварительная компиляция — перед исполнением модули, содержащие текст на встроенном языке, преобразуются во внутренний код;
- кэширование скомпилированных модулей в памяти;
- мягкая типизация — тип переменной определяется типом значения, которое она содержит, и может изменяться в процессе работы;
- Встроенный язык 1С является предметно-ориентированным языком. Обладает некоторыми возможностями объектно-ориентированных языков: обращение к свойствам и методам объектов. Но программно нельзя создавать новые классы, только визуально.
- Динамическая типизация переменных — тип переменной определяется в процессе выполнения программы. Одна переменная может принимать значения разных типов.

В частности, все операторы языка имеют как русское, так и англоязычное написание, которое можно использовать одновременно в одном исходном тексте.

*Например:*

```
Message("Hello, World!");  
Сообщить("Привет, Мир!");
```

#### 2. Комментарии

В тексте модуля все что идет после двойного слеша «//» является комментарием. При компиляции программного модуля комментарии игнорируются.

Комментарии нужны чтобы сделать код более понятным для программиста. Как правило на уровне процедуры или функции комментарии отвечают на вопрос «Что?» (что делает данная функция). Внутри процедуры на вопрос «Как?» (алгоритм функции). На уровне строки кода на вопрос «Почему?» (почему переменной было присвоено такое-то значение и т.п.)

```
//это комментарий  
a = 2; // это тоже комментарий
```

### 3. Операторы встроенного языка

Это основной блок программного модуля. Между собой разделяются точкой с запятой «;». На одной строке можно расположить несколько операторов. Также один оператор может быть на нескольких строках (тогда в конце первой строки не нужно ставить «;»).

Операторы бывают двух видов:

- Операторы объявления переменных
- Исполняемые операторы, которые манипулируют с переменными

```
Перем Переменная; //это оператор объявления переменной
```

```
Переменная = 1; // исполняемый оператор
```

```
МояПроцедура(Переменная); // тоже исполняемый оператор
```

При написании кода на встроенном языке можно писать на русском, на английском и даже смешивать.

```
Перем Переменная;  
Переменная = Undefined;
```

Регистр не имеет значения:

```
Переменная = 1;  
переменная = 1;  
переменная = 2; //это все одна и та же переменная
```

### 4. Виды и структура программных модулей

**Модулем** называется программа на встроенном языке 1С:Предприятие. Модули располагаются в заданных точках конфигурации и вызываются для выполнения в заранее известные моменты работы системы 1С:Предприятие.

В программных модулях содержится исполняемый код на языке 1С, который необходим для того, чтобы определенным образом реагировать на действия системы или пользователя, когда визуальных средств разработки недостаточно. Также в программных модулях можно описывать собственные методы (процедуры и функции).

Программные модули располагаются в тех местах конфигурации, которые могут требовать описания специфических алгоритмов функционирования. Эти алгоритмы следует оформлять в виде процедур или функций, которые будут вызываться самой системой в заранее предусмотренных ситуациях (например, при открытии формы справочника, при нажатии кнопки в диалоговом окне, при изменении объекта и т.д.).

Каждый отдельный программный модуль воспринимается системой как единое целое, поэтому все процедуры и функции программного модуля выполняются в едином контексте.

Контекст выполнения модулей делится на:

Клиентский

Серверный

Кроме того, некоторые программные модули могут быть скомпилированы как на стороне клиента, так и на стороне сервера.

Обычно программный модуль состоит из **трех** разделов:

Раздел описания  
переменных

Раздел процедур и  
функций

Раздел основной  
программы

**Раздел описания переменных** размещается от начала текста модуля до первого оператора *Процедура* или оператора *Функция* или любого исполняемого оператора. В этом разделе могут находиться только операторы объявления переменных *Перем*.

**Раздел процедур и функций** размещается от первого оператора *Процедура* или оператора *Функция* до любого исполняемого оператора вне тела описания процедур или функций.

**Раздел основной программы** размещается от первого исполняемого оператора вне тела процедур или функций до конца модуля. В этом разделе могут находиться только исполняемые операторы. Данный раздел выполняется в момент инициализации модуля. Обычно в разделе основной программы имеет смысл размещать операторы инициализации переменных какими-либо конкретными значениями, которые необходимо присвоить до первого вызова процедур или функций модуля.

```
//*****РАЗДЕЛ ОПИСАНИЯ ПЕРЕМЕННЫХ *****

Перем фамилия Экспорт;    //это глобальная переменная
Перем Имя, Отчество;      //это переменная модуля
Перем ФИО;                //это тоже переменная модуля и к ней можно обращаться
                           //из любой процедуры и функции модуля

//***** РАЗДЕЛ ПРОЦЕДУР И ФУНКЦИЙ *****

Э Процедура Процедурал ( )
    Перем Итог;              //Итог это локальная переменная (переменная процедуры)
    Итог = фамилия+ " "+Имя+ " "+Отчество;
КонецПроцедуры

Э функция функция1 ( )
    // операторы функции
    Возврат(фамилия + " "+ Имя);
Конецфункции

//*****РАЗДЕЛ ОСНОВНОЙ ПРОГРАММЫ *****

фамилия ="Иванов";
Имя = "Иван";
Отчество = "Иванович";
//*****
```

В конфигурации существует несколько видов модулей:

- модуль приложения;
- модуль внешнего соединения;
- общие модули;
- модуль сеанса;
- модули форм;
- модули объектов конфигурации;
- модуль менеджера объектов;
- модуль команды.

Для написания и редактирования текстов программных модулей предназначен редактор текстов и модулей.

**Модуль приложения (управляемого или обычного)**

- может содержать все 3 раздела;
- выполняется на стороне клиента;
- располагается в корневом разделе конфигурации.

В модуле приложения описываются процедуры (обработчики) событий, которые инициализируются при старте и окончании работы системы. Например, при начале работы приложения можно обновить какие-либо данные конфигурации, а при завершении работы – поинтересоваться, стоит ли вообще выходить из программы. Кроме того, в данном модуле перехватываются события от внешнего оборудования, например, торгового или фискального. Стоит отметить, что модуль приложения выполняется только в случае интерактивного запуска приложения, то есть когда запускается окно программы. Этого не происходит, если приложение запускается в режиме com- соединения.

#### **Модуль внешнего соединения**

- может содержать все 3 раздела;
- выполняется на стороне сервера;
- располагается в корневом разделе конфигурации.

Назначение модуля аналогично назначению модуля приложения. В нем идет обработка событий старта и завершения работы приложения. Модуль внешнего соединения срабатывает, когда запуск приложения происходит в режиме com-соединения. Сам процесс внешнего соединения – это процесс не интерактивный. В этом режиме происходит программная работа с информационной базой и не происходит открытия окна приложения, что накладывает определенные ограничения на использование методов, предназначенных для интерактивной работы. В этом режиме нельзя использовать вызовы диалоговых форм, предупреждений и сообщений пользователю и т.п. Они просто не будут выполняться.

Главное отличие от модуля приложения заключается в том, что в режиме com-соединения вся работа с информационной базой происходит на стороне сервера, поэтому модуль внешнего соединения компилируется на стороне сервера.

#### **Модуль сеанса**

- может содержать раздел процедур и функций;
- выполняется на стороне сервера;
- располагается в корневом разделе конфигурации.

Это узкоспециализированный модуль, предназначенный исключительно для инициализации параметров сеанса. Его использование обусловлено тем, что само приложение может запускаться в различных режимах (что приводит к выполнению либо модуля управляемого, либо обычного приложения, либо модуля внешнего соединения), а инициализацию параметров сеанса необходимо производить вне зависимости от режима запуска. Чтобы не писать один и тот же программный код во всех трех указанных модулях, и потребовался дополнительный модуль, который выполняется вне зависимости от режима запуска приложения.

В модуле сеанса существует одно единственное событие «УстановкаПараметровСеанса», которое выполняется самым первым, даже раньше события модуля приложения ПередНачаломРаботыСистемы. В нем не доступны раздел объявления переменных и раздел основной программы. А так же нельзя объявлять экспортные методы. Модуль компилируется на стороне сервера.

#### **Общие модули**

- может содержать раздел процедур и функций;
- выполняется на стороне сервера или клиента (зависит от настроек модуля);
- располагается в ветке дерева объектов конфигурации «Общие» → «Общие модули»;



Общие модули предназначены для описания некоторых общих алгоритмов, которые будут вызываться из других модулей конфигурации. Общий модуль не содержит областей объявления переменных и основного текста программы. В нем можно объявлять экспортные методы, доступность которых будет определяться настройками модуля (на какой стороне он выполняется: на стороне сервера или клиента). В связи с тем, что раздел описания переменных не доступен, определять глобальные переменные в общих модулях нельзя. Для этого можно использовать модуль приложения.

Поведение общего модуля зависит от выставленных параметров (глобальный или нет, различные флаги компиляции, доступен ли вызов сервера и т.д.).

### **Модуль формы**

- может содержать все 3 раздела;
- выполняется на стороне сервера и клиента.

Модуль формы предназначен для обработки действий пользователя с данной формой (обработка события нажатия кнопки, изменения реквизита формы и т.д.). Так же существуют события связанные непосредственно с самой формой (например, ее открытие или закрытие).

Структура управляемой формы содержит раздел объявления переменных, описания процедур и функций и основной текст программы (выполняется в момент инициализации формы). К стандартным событиям формы можно обратиться через список ожидаемых процедур и функций формы (**Ctrl+Alt+P**), либо через палитру свойств самой формы.

### **Модуль объекта**

- может содержать все 3 раздела;
- выполняется на стороне сервера.

Данный модуль имеется у большинства объектов конфигурации и предназначен, в общем случае, для обработки событий, непосредственно связанных с объектом. Например, события записи и удаления объектов, проверка заполнения реквизитов объекта, проведение документа и т.д.

События модуля формы будут выполняться исключительно в конкретной форме объекта, то есть при открытии конкретной формы. А события модуля объекта будут вызываться в любом случае, даже в момент программной работы с объектом. Поэтому, если необходимо методы связанные с объектом без привязки к конкретной форме объекта, то лучше использовать для этого модуль объекта.

### **Модуль менеджера объекта**

- может содержать все 3 раздела;
- выполняется на стороне сервера.

Модуль менеджера объектов появился только начиная с версии 1С 8.2. Модуль менеджера существует у всех прикладных объектов и предназначен для управления этим объектом как объектом конфигурации. Модуль менеджера позволяет расширить функциональность объекта за счет введения (написания) процедур и функций, которые относятся не к конкретному экземпляру объекта базы данных, а к самому объекту конфигурации. Модуль менеджера объектов позволяет размещать общие процедуры и функции для данного объекта и обращаться к ним из вне, например, из обработки (конечно, если эта процедура или функция будет с ключевым словом Экспорт). Что это нам дает нового? В общем-то, ничего, кроме упорядочивания процедур по объектам и хранения их в отдельных местах - Модулях менеджеров объектов. Мы можем с таким же успехом эти процедуры и функции помещать в общих модулях, но 1С советует общие процедуры и функции объектов размещать в Модуле менеджера объектов. Примеры использования процедур и функций Модуля менеджеров объектов: первоначальное заполнение отдельных

реквизитов справочника или документа по определенным условиям, проверка заполнения реквизитов справочника или документа по определенным условиям и т.д.

#### **Модуль команды**

- может содержать раздел описания процедур и функций
- выполняется на стороне клиента

Команды – это объекты, подчиненные прикладным объектам или конфигурации в целом. У каждой команды есть модуль команды, в котором можно описать предопределенную процедуру `ОбработкаКоманды()` для выполнения этой команды

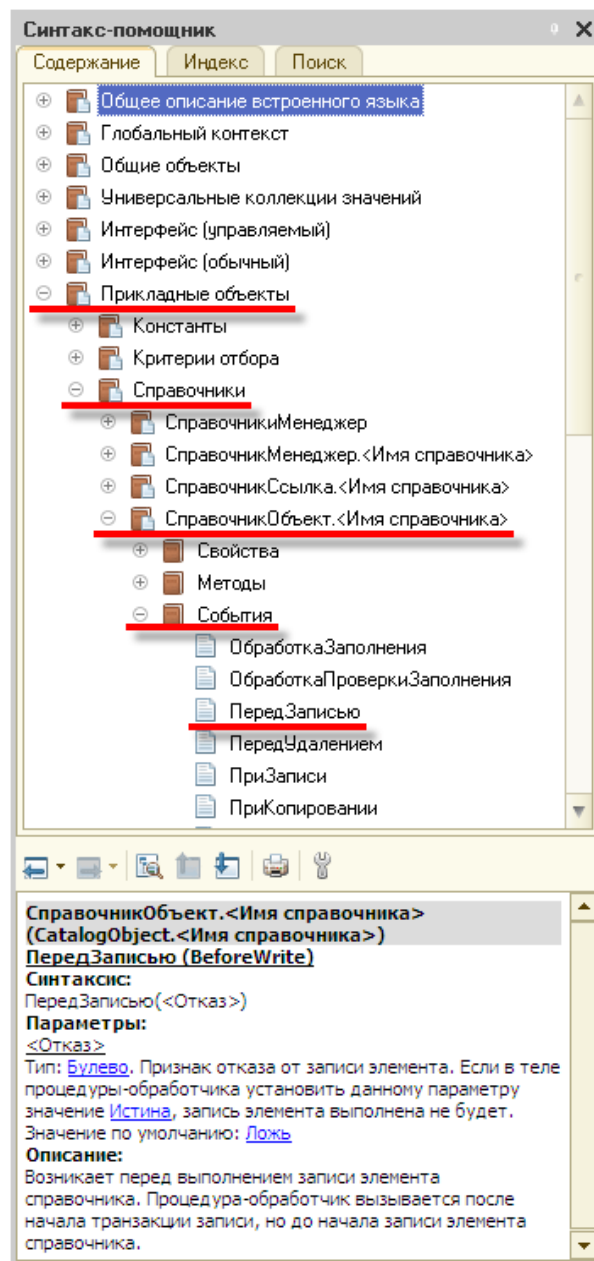
#### **5. Событийная ориентированность встроенного языка**

В отличие от других языков программирования в 1С нет метода `Main` с которого начинается выполнение программы. Модули выполняются в определенных ситуациях, которые заранее известны и называются событиями.

Назначение встроенного языка в системе 1С:Предприятие определяется идеологией создания прикладных решений. Прикладные решения в 1С:Предприятии 8 не кодируются целиком. Большая часть прикладного решения создается разработчиком путем визуального конструирования — создания новых объектов конфигурации, задания их свойств, форм представления, взаимосвязей и пр. Встроенный язык используется лишь для того, чтобы определить поведение объектов прикладного решения, отличное от типового, и создать собственные алгоритмы обработки данных.

По этой причине модули, содержащие текст на встроенном языке, используются системой в конкретных, заранее известных ситуациях, которые могут возникнуть в процессе работы прикладного решения. Такие ситуации называются событиями. События могут быть связаны с функционированием объектов прикладного решения или с самим прикладным решением, как таковым.

Например, с функционированием объекта прикладного решения Справочник связан ряд событий, среди которых есть событие `ПередЗаписью`:



Это событие возникает непосредственно перед тем, как данные элемента справочника должны быть записаны в базу данных. Разработчик, используя встроенный язык, может описать алгоритм, который, например, будет проверять корректность данных, введенных пользователем. Разместив этот алгоритм в соответствующем модуле, разработчик обеспечит то, что каждый раз, как пользователь будет выполнять запись элемента справочника, система будет выполнять созданный разработчиком алгоритм и проверять, не забыл ли пользователь заполнить обязательные реквизиты справочника.

Таким образом можно сказать, что встроенный язык является скриптовым языком для программирования бизнес-логики, а использование модулей на встроенном языке является событийно-зависимым, т. е. выполнение модулей происходит при возникновении определенных событий в процессе функционирования прикладного решения.

## 6. Примитивные типы данных

Примитивные типы данных — эти типы не являются чем-то особенным для 1С:Предприятия 8. Как правило, такие типы данных существуют и в других программных системах.

Простые (примитивные) типы данных:

- строковые константы;
- числовые выражения;
- булевские значения (Истина и Ложь);
- тип Дата;
- тип данных с единственным значением Неопределено;
- тип данных NULL, который также состоит из единственного значения;
- тип данных типа Тип.

Строковые константы. Тип данных Строка (строковая константа) состоит из различных символов. Значения данного типа содержат строку в формате Unicode произвольной длины. Строка всегда обрамляется кавычками.

*Например:*

Сообщение.Текст = “Присутствуют незаполненные данные”;

Т.е. строка “Присутствуют незаполненные данные” присваивается реквизиту Текст объекта Сообщение. Все, что обрамлено в кавычки, считается строкой.

Строка может состоять из любых символов. Строки могут быть многострочными. При этом каждую новую строку необходимо определять в кавычки.

*Например:*

Текст = “Неверно заполнен реквизит” “Проведение документа невозможно”;

Точка с запятой ставится только в конце последней строки.

Существует еще один способ – весь текст обрамлять только в одни кавычки, но каждая новая строка должна начинаться с вертикальной полосы. Такой синтаксис наиболее часто используется в типовых конфигурациях. В частности, в языке запросов.

*Например:*

Запрос.Текст = «ВЫБРАТЬ

Сотрудники.Наименование КАК Сотрудник,

Сотрудники.ДатаРождения КАК ДатаРождения

ИЗ

Справочник.Сотрудники КАК Сотрудники

ГДЕ

НЕ Сотрудники.ЭтоГруппа»;

Следует отметить, что для строк определена операция сложения. Это не арифметическая операция, ее называют операцией **конкатенации**. Т.е. нужно объединить, например, две строки, при этом между строками ставится знак сложения «+»:

*Например:*

ПолноеНаименование = "Вентилятор "+"бытовой";

В итоге ПолноеНаименование будет иметь значение "Вентилятор бытовой".

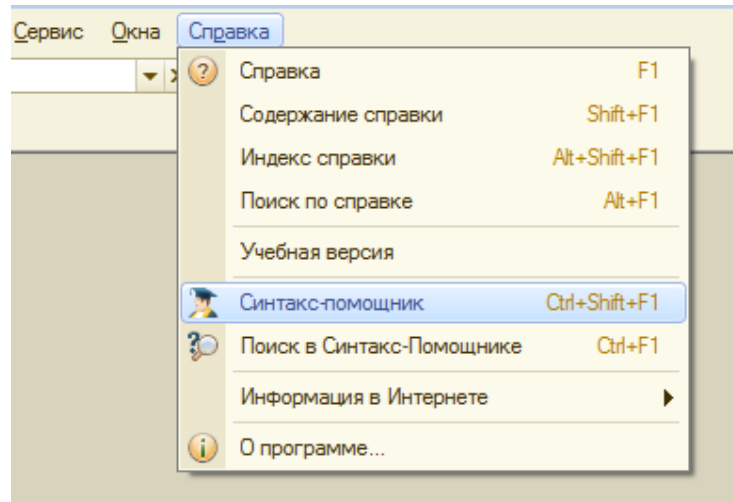
Таким образом, происходит склеивание строк. Операция конкатенации, естественно, применима и для большего количества строк. Другие операции (вычитание, умножение, деление) для строк не допустимы. Если внутри строки какое-то слово нужно обрамлять в кавычки, то кавычку внутри строки нужно определять двойной кавычкой.

*Например:*

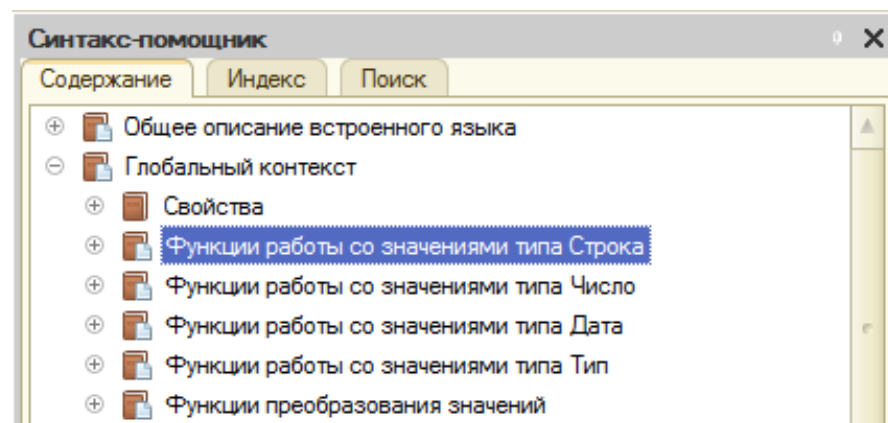
Текст = “Ошибка в модуле “”Общий модуль1”””;

Над строками возможны различные операции преобразования строк, определение нескольких первых левых символов, определение нескольких крайних правых символов, поиск подстроки внутри строки и т.д.

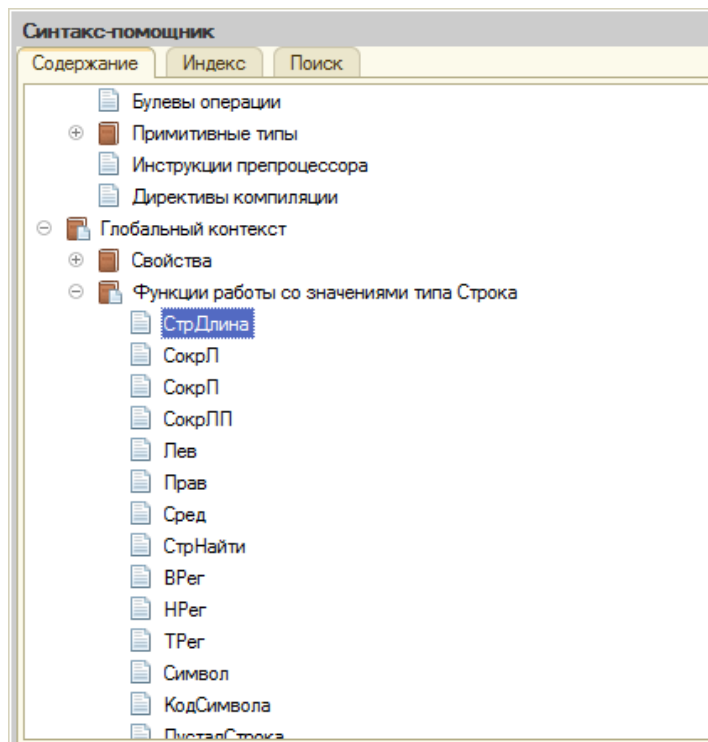
Все эти функции доступны в любом месте конфигурации. В Синтакс-помощнике



они находятся в разделе Глобальный контекст → Функции работы со значениями типа Строка.



Функций достаточно большое количество и их обычно достаточно для работы со строковыми константами.



*Например:*

```
Сообщить(СтрДлина("123456789")); // 9
```

**Числовые выражения.** Числовыми могут быть переменные модулей и реквизиты объектов базы данных. Для числа существует ограничение разрядности. Для числового реквизита длина целой части не может превышать 32 символов. В качестве разделителя целой и дробной части используется "."

Точность дробной части не может превышать 10 цифр. Когда описывается переменная и присваивается ей числовое значение, то нигде не фиксируется ее разрядность. Тем не менее, для переменных тоже существуют ограничения.

Максимально допустимая разрядность для числа – это 38 знаков. Такое ограничение не препятствует решению любых экономических задач, т.е. любую денежную величину можно описать с помощью этих чисел. Тем не менее, если все-таки потребуется описать большие величины для решения каких-то математических задач, то в теории программирования есть алгоритмы, позволяющие описать числа с любой размерностью на основании существующих ограничений.

Операции, применимые для чисел:

– обычные арифметические операции (-, +, \*, /). У умножения и деления приоритет больше, чем у сложения и вычитания. Скобки имеют наивысший приоритет.

*Например:*

<p><math>A = (2+7) * 3;</math> в результате <math>A=27</math></p> <p><math>B = 2 * ((2+7) - 4);</math> в результате <math>B = 10</math></p>	<p>// Приоритет выполнения арифметических операций:</p> <p><math>A =</math> Сначала вычисляем <math>2 + 7</math>, затем полученное значение умножаем на <math>3</math>;</p> <p><math>B =</math> Сначала вычисляем <math>2 + 7</math>, затем от полученного значения отнимаем <math>4</math>, затем полученное значение умножаем на <math>2</math>.</p>
---	--

Есть еще унарные операции «+» и «-», у которых приоритет идет сразу за скобками;

– операция “остаток от деления” (%).

Например:

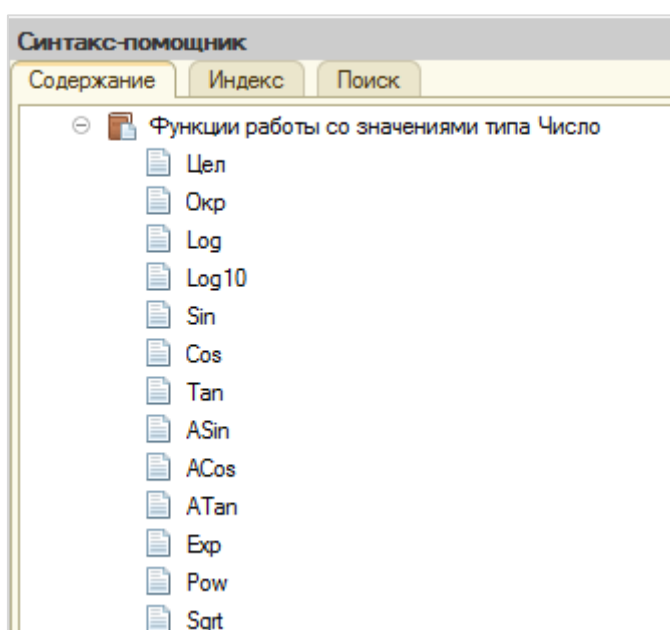
$12 \% 5 = 2$ ;

– математические функции, которые можно применять для чисел (тригонометрические функции, возведение в степень, извлечение квадратного корня, округление до указанной разрядности, выбор целой части числа).

```
// Округлим цену до копеек
ОкругленнаяЦена = Окр(Цена, 2);

// Округлим цену до сотен рублей
ОкругленнаяЦена = Окр(Цена, -2);
```

Математические функции для чисел можно посмотреть в Синтакс-помощнике:



Например:

Сообщить(Pow(10, 3)); //  $10^3 = 1000$

Сообщить(Sqrt(25)); // 5

Булевские значения. Существует только два значения *Истина* и *Ложь*, которые могут быть получены различными способами. Можно, например, использовать операции сравнения чисел или дат. В итоге будет получаться некое булевское значение, которое в дальнейшем наиболее часто используется в условных операторах и в операторах цикла.

Применяется в основном для формирования результата логического выражения.

Тип данных Дата предназначен для работы с датами (дата документа, дата события и пр.) Данный тип содержит не только значение самой даты, но и значение времени в пределах этой даты (часы, минуты и секунды)

Значения данного типа содержит дату григорианского календаря (с 01 января 0001 года) и время с точностью до 0,1 миллисекунды.

Для описания даты существует два способа.

1) С использованием литерала. Литерал пишется в одинарных кавычках. Сначала пишется год, потом месяц и затем день. При необходимости, можно указать и время, т.к. в системе 1С:Предприятие 8 любая дата содержит в себе и дату и время.

*Например:*

ДатаДокумента = '20150315121020';

Если время не указано, то по умолчанию оно равно нулю.

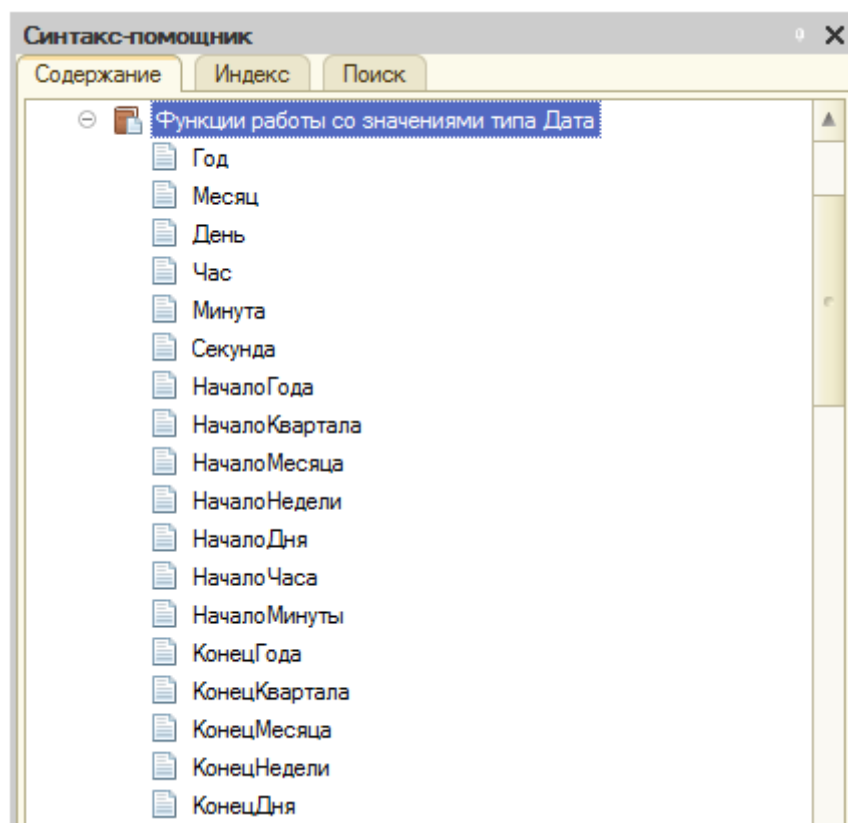
В описании даты можно использовать любой разделитель.

*Например:*

ДатаДокумента = '2015.03.15';

2) С использованием функции глобального контекста *Дата* (<Значение>). В этом случае передается в качестве параметров этой функции год, месяц, день через запятую.

Функции для работы со значением типа Дата можно посмотреть в Синтаксис-помощнике:



*Например:*

Дата = '20160107125905'; // 7 января 2016 года 12:59:05

Сообщить(ДеньГода(Дата)); // 7

Сообщить(ДеньНедели(Дата)); // 4 т.е. четверг (нумерация с понедельника)

Сообщить(НачалоГода(Дата)); // 01.01.2016 0:00:00

Сообщить(КонецГода(Дата)); // 31.12.2015 23:59:59

Значения типа NULL и Неопределено. Значение типа Неопределено применяются, когда необходимо использовать пустое значение, не принадлежащее ни к одному другому типу.



Данный тип (Неопределено) появляется, во-первых, если есть некая переменная, и она не инициализирована (тип данных не определен).

Второй пример: тип данных Неопределено возвращают многие функции встроенного языка, если действие не может быть выполнено.

Например, поиск элемента справочника по наименованию в том случае, если у какого-либо справочника нет такого наименования элемента. Метод НайтиПоНаименованию будет возвращать значение Неопределено.

Для написания Неопределено не нужно использовать никаких кавычек, запятых, скобок и т.д. Если имеется список документов, и этот список пустой (в нем, соответственно нет ни одной строки), то текущая строка будет принимать значение Неопределено. Если в информационной базе есть реквизит с составным типом данных, то пустое значение данного реквизита будет равно Неопределено.

NULL обозначает *отсутствие* значения, т.е. в буквальном смысле пустое место. В этом и состоит его отличие от типа Неопределено.

NULL – это тип данных, который получается при работе с запросами. Применяется при работе с БД (при соединении таблиц), используется для определения отсутствующего значения при работе с БД.

ВЫБРАТЬ

Справочник.Номенклатура.Наименование,

Справочник.Номенклатура.ЗакупочнаяЦена

ГДЕ

Справочник.Номенклатура.ЗакупочнаяЦена Есть NULL

Тип данных Тип. Основное применение этого типа данных заключается в том, чтобы сравнить значение некой переменной или реквизита базы данных с конкретным типом. Т.е. в алгоритме нужно понять, какой тип у данного объекта. Примечательно, что этот тип данных не имеет литерала.

Функции для работы со значением типа Тип можно посмотреть в Синтакс-помощнике:

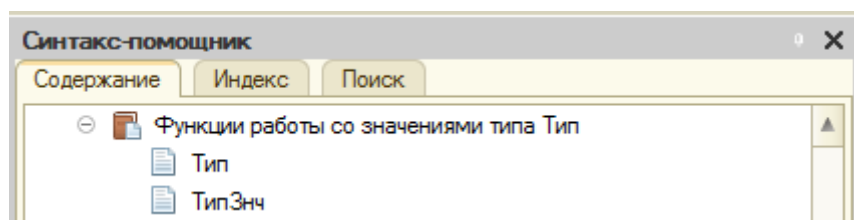


Рисунок 2.7 – Функции для работы со значением типа Тип

Для того, чтобы получить тип некоторого объекта (это может быть переменная, либо реквизит базы данных, либо реквизит формы), используется функция ТипЗнч. В эту функцию передается тот объект, для которого требуется получить тип данных. В качестве возвращаемого значения эта функция возвращает именно тип типа Тип. В дальнейшем следует его сравнить с каким-либо интересующим типом.

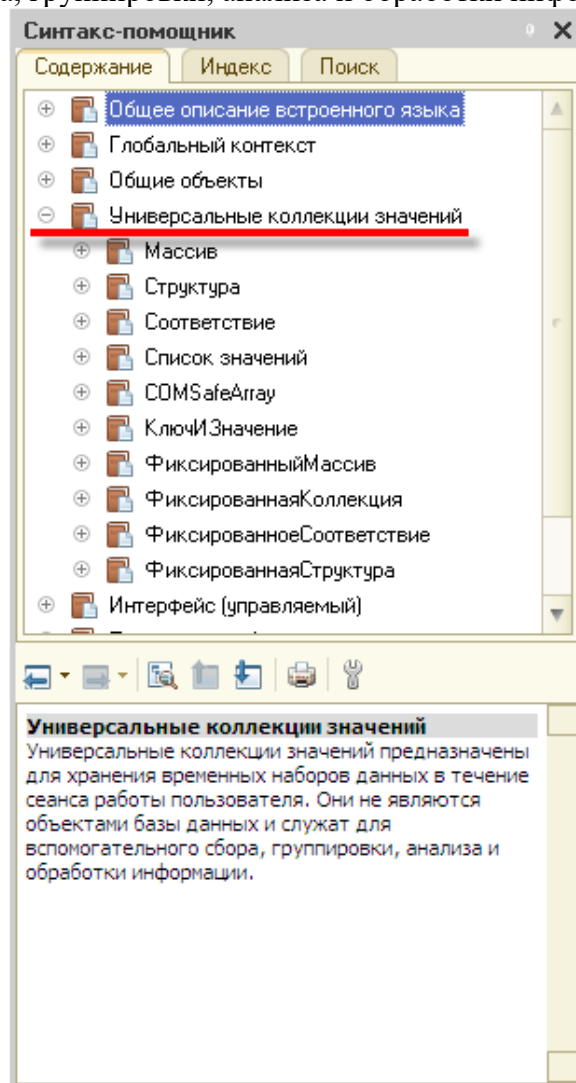
Например:

Если ТипЗнч(Элемент) = Тип (“СправочникСсылка.Номенклатура”) Тогда  
Сообщить (“Это товар”);  
КонецЕсли;

## 7. Сложные типы данных. Универсальные коллекции значений

Встроенный язык поддерживает работу с большим количеством разнообразных объектов. Безусловно, основную группу объектов составляют прикладные объекты, позволяющие описывать алгоритмы функционирования бизнес-логики.

Однако не менее важной группой являются объекты, предназначенные для хранения временных наборов данных в течение сеанса работы пользователя. Как правило, они служат для вспомогательного сбора, группировки, анализа и обработки информации:



Коллекция значений – это некий контейнер, в котором может содержаться обычно любое количество элементов. При этом каких-либо жестких ограничений на тип данных зачастую не накладывается.

**Типы данных** Универсальные коллекции – список (набор) объектов данных любых типов, к значениям которого можно обратиться перебором или по указанному индексу (ключу). Нумерация элементов коллекций начинается с 0.

Почти любую универсальную коллекцию можно создать с помощью конструктора (исключением являются табличные части, которые выступают в качестве объектов конфигурации).

Коллекция может создаваться в результате работы какой-либо функции (функция возвращает в качестве значения универсальную коллекцию). Можно получить новую коллекцию вручную, обратившись к конструктору и создав экземпляр класса.

**Массив.** Представляет собой пронумерованную коллекцию значений произвольного типа. К элементу массива можно обращаться по его индексу. В качестве элементов массива

могут выступать, в частности, другие массивы. Это позволяет создавать многомерные массивы.

Например:

НашМассив = Новый Массив;

Для всех коллекций используется обход элементов коллекции. Обход возможен двумя способами: циклом Для и циклом Для каждого из.

```
НашМассив = Новый Массив;  
  
// Использование цикла Для  
КоличествоВМассиве = НашМассив.Количество();  
Для Индекс=0 По КоличествоВМассиве-1 Цикл  
  
    Сообщить(НашМассив[Индекс]);  
  
КонецЦикла;  
  
// Использование цикла Для каждого  
Для каждого ЭлементМассива Из НашМассив Цикл  
  
    Сообщить(ЭлементМассива);  
  
КонецЦикла;
```

Для большинства универсальных коллекций применимы методы:



КОЛИЧЕСТВО — это функция, которая возвращает количество элементов коллекции.

Метод ИНДЕКС существует не у всех коллекций, а только у тех, на элементы которой можно сослаться. В качестве примера можно привести ТаблицуЗначений. Метод Индекс позволяет определить, какой индекс соответствует данной строке (т.е. текущую позицию строки в таблице). **Значения индекса начинаются с 0.**

Методы добавления новых значений в данную коллекцию существуют практически у любой универсальной коллекции.

Для того, чтобы **добавить** элемент в Массив можно использовать метод ДОБАВИТЬ, в скобках указать добавляемое значение. При этом значение будет добавляться в конец списка, т.е. Массив будет постоянно увеличиваться за счет последней позиции.

Например, массив можно заполнить значениями от 0 до 10 с использованием метода Добавить.

```
НашМассив = Новый Массив;  
  
// Первый способ  
Для Число=0 По 10 Цикл  
  
    НашМассив.Добавить(Число);  
  
КонецЦикла;
```

Другой метод, который позволяет добавлять значения в коллекцию – метод ВСТАВИТЬ. Он отличается от метода Добавить тем, что можно указать, в какое место нужно вставить добавляемый элемент.

*Синтаксис:*

Вставить (<Индекс>, <Значение>)

Первым параметром указывается индекс, в который будет вставлено новое значение. Например, можно указать, что каждое значение нужно вставлять в начало списка.

На рисунке представлено, как заполнить Массив значениями от 0 до 10 с использованием метода Вставить.

```
// Второй способ  
Для Число=0 По 10 Цикл  
  
    НашМассив.Вставить(0, Число);  
  
КонецЦикла;
```

Для удаления элементов из коллекции используется метод УДАЛИТЬ. В методе Удалить указывается по индексу, какой элемент необходимо удалить.

*Синтаксис:*

Удалить(<Индекс>)

*Например:*

НашМассив.Удалить(5)

Практически у всех коллекций существует метод поиска значения – НАЙТИ. В некоторых коллекциях можно поставить какие-либо ограничения. Например, в ТаблицеЗначений можно указать те строки, те колонки, в которых нужно осуществлять поиск. Если значение найдено, то данный метод возвращает индекс или определенную строку. Если значение не найдено, возвращается значение типа Неопределено. Применительно к Массиву возвращается Индекс, либо значение Неопределено.

*Например:*

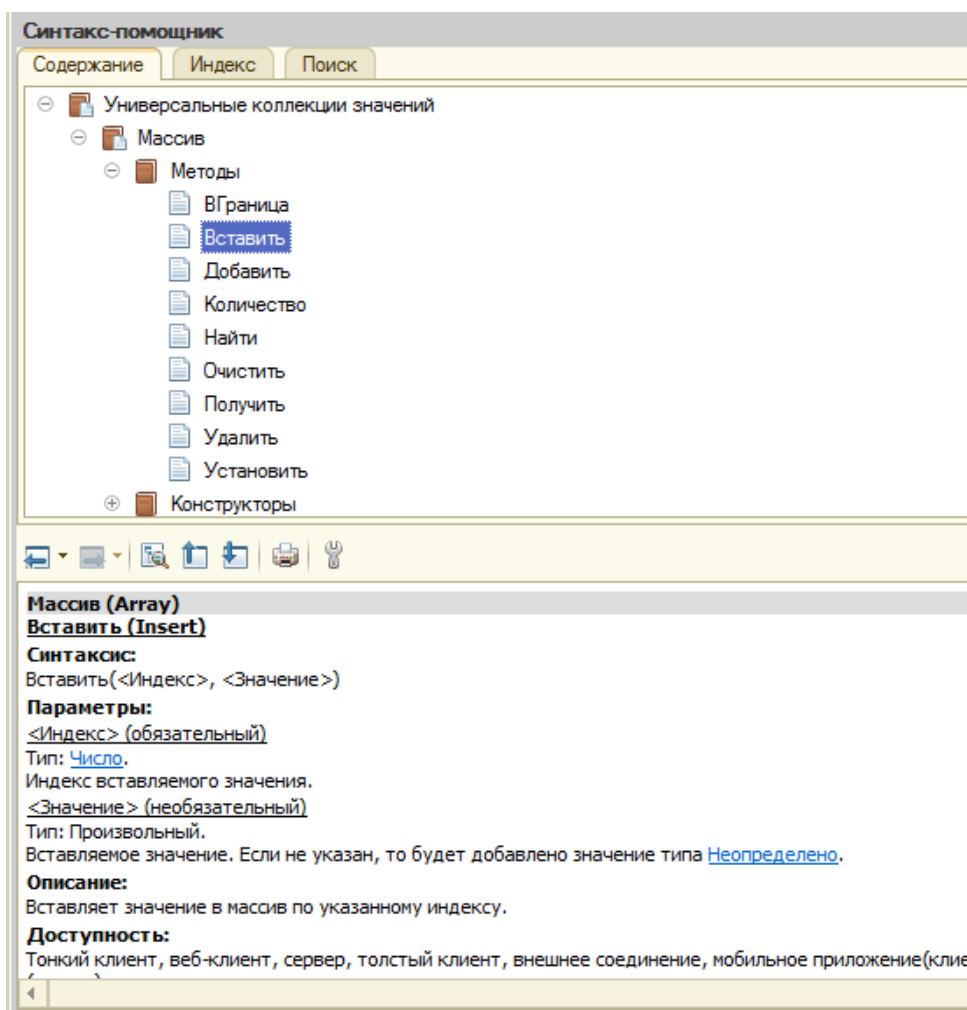
НашаПеременная = НашМассив.Найти(8);

Универсальные коллекции можно очень быстро очищать, т.е. удалить абсолютно все элементы. Для этого используется метод ОЧИСТИТЬ(), который удаляет элементы Массива, строки ТаблицыЗначений, либо данные других коллекций.

Для того чтобы обратиться к элементу Массива *НашМассив* (см. пример выше), можно использовать обращение по индексу, для этого индекс указывается в квадратных

скобках. Например, *НашМассив[3]*. Обратите внимание, в этом случае система возвращает элемент Массива с индексом 3, а по порядку это четвертый элемент Массива.

Перечень методов Массива можно просмотреть в Синтакс-помощнике.



**Структура.** Представляет собой поименованную коллекцию, состоящую из пар Ключ - Значение.

**Ключ** – это строго строковый тип данных, который описывает значение. Например, Ключу «Код» может соответствовать значение 113; Ключу «Имя» значение «Вася». На само **Значение** ограничение типа данных не накладывается.

К элементу структуры можно обращаться по значению его ключа, т.е. по имени. Обычно используется для хранения небольшого количества значений, каждое из которых имеет некоторое уникальное имя.

Если Структура называется НашаСтруктура, то обращаться к ее двум значениям необходимо следующим образом:

НашаСтруктура.Код  
НашаСтруктура.Имя

Отдельно следует отметить, что в качестве Ключа в Структуре может выступать не абсолютно любая строка. Накладываются определенные ограничения. Ключ должен выступать в качестве идентификатора. Это означает, что в Ключе не должно быть пробелов, и он не может начинаться с цифры.

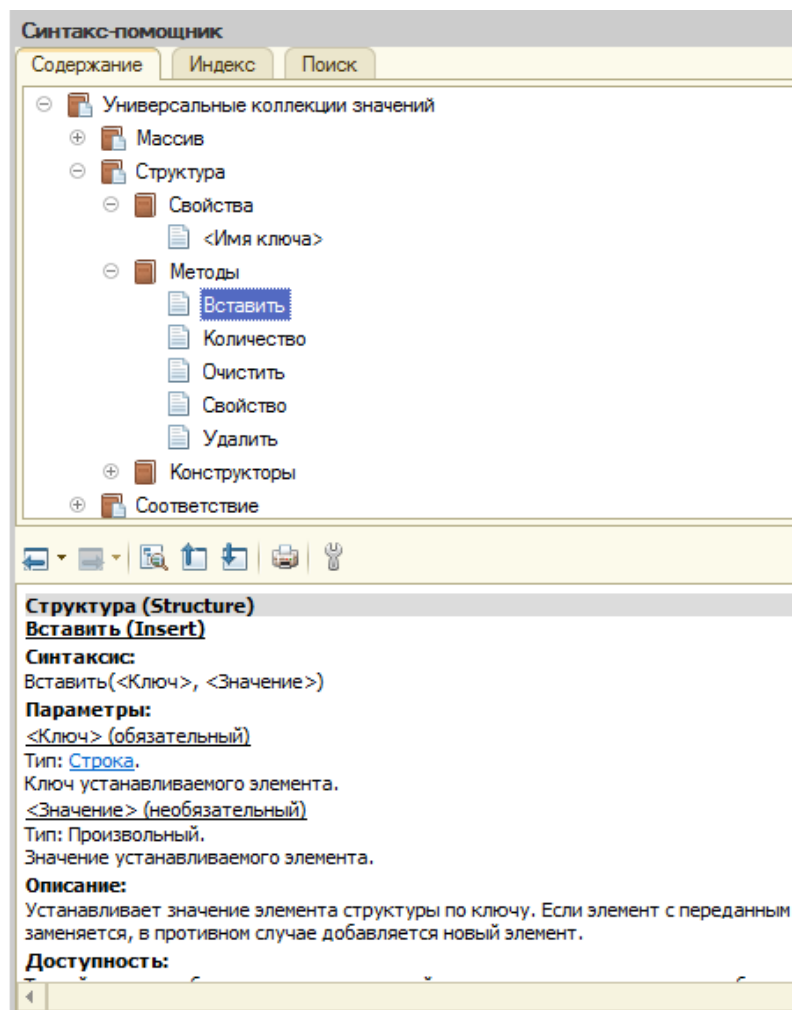
В отличие от Массива, где можно просто указать количество элементов для всех размерностей, в Структуре возможно задавать само содержание.

*Например:*

НашаСтруктура = Новый Структура (“Код,Имя”, 133, “Вася”);

Через запятую перечисляются сначала имена Ключей, а потом, соответственно, в той же последовательности значения параметров.

Для Структуры существует метод ВСТАВИТЬ.



*Например:*

НашаСтруктура.Вставить(“ЧленовСемьи”,3);

**Структура.** Представляет собой поименованную коллекцию, состоящую из пар ключ — значение. Ключ может быть только строковым, значение — произвольного типа. К элементу структуры можно обращаться по значению его ключа, т. е. по имени. Обычно используется для хранения небольшого количества значений, каждое из которых имеет некоторое уникальное имя.

**Соответствие.** Также как и структура, представляет собой коллекцию пар ключ — значение. Однако, в отличие от структуры, ключ может быть практически любого типа.

```
НашеСоответствие = Новый Соответствие;
НашеСоответствие.Вставить(НачалоДня(ТекущаяДата()), "Вторник");
НашеСоответствие.Вставить(НачалоДня(ТекущаяДата())-24*3600, "Понедельник");
Сообщить(НашеСоответствие[НачалоДня(ТекущаяДата())]);
```

Соответствие удобно применять тогда, когда необходимо связать какие-либо две структуры. Например, каждой строке табличной части необходимо сопоставить строку из таблицы значений. В этом случае в качестве ключа Соответствия используется строка табличной части и указывается соответствующее значение.

При вставке элементов в коллекцию Соответствие помимо метода ВСТАВИТЬ(<Ключ>,<Значение>) существует другой способ вставки значения – это использование обычного оператора присваивания.

*Например:*

```
НашеСоответствие = Новый Соответствие;
Соответствие[777] = 999;
```

Т.е. если элемент в коллекции не присутствовал, то с помощью оператора присваивания он будет добавлен, а если присутствовал, то будет обновлен. Это является отличием от Структуры.

Список значений. Используется, как правило, для решения интерфейсных задач. Позволяет строить динамические наборы значений и манипулировать ими (добавлять, редактировать, удалять элементы, сортировать). Он может содержать значения любого типа, кроме того, в одном списке типы хранимых значений могут быть разными.

Например, список значений может использоваться для выбора конкретного документа из списка возможных документов, сформированного по сложному алгоритму.

Список Значений представляет собой линейный список элементов любого типа данных. Каждый элемент состоит из нескольких значений. Схематично список значений можно представить в виде списка с четырьмя колонками.

Первая колонка – Отметка. Она имеет булевский тип данных и позволяет пользователю либо ставить флажки, либо их снимать.

Вторая колонка – это картинка, которая может каким-то образом визуальнo изображать данный элемент, т.е. ставить в соответствие данной строке какую-либо картинку.

Третья колонка – само хранимое значение, т.е. это любой тип данных, причем в разных строках он может быть различным.

Четвертая колонка – это представление, т.е. это некое строковое описание данного значения. Представление будет выводиться пользователю, когда он будет просматривать данный элемент.

| Процедура УзнатьОПодарке()

```
// Создаем список значений
```

```
СписокЗначений = Новый СписокЗначений;
СписокЗначений.Добавить("Торт", "Торт", Ложь,);
СписокЗначений.Добавить("ДенежнаяПремия", "500 рублей", Ложь,);
СписокЗначений.Добавить("ПочетнаяГрамота", "Почетная грамота", Ложь,);
ВсегоВыбрано = 0;
```

Список Значений – это тот объект, с которым может визуальнo работать пользователь. Т.е. Список Значений можно вывести на форму.

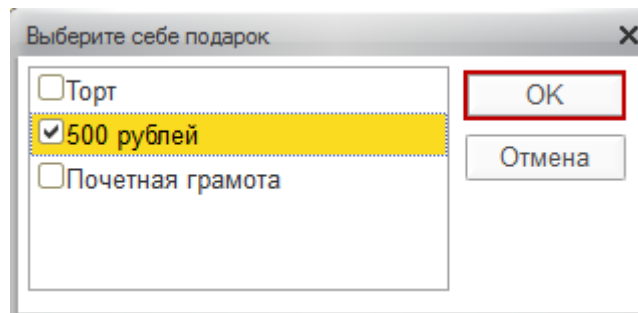


Таблица значений. Таблица значений позволяет строить динамические наборы значений и манипулировать ими. Она может быть наполнена значениями любого типа, и в одной таблице типы хранимых значений могут быть разными.

Одним из примеров использования таблицы значений может служить организация представления в форме списка элементов справочника, отображенных по сложному алгоритму.

#### Отличия Таблицы Значений от двухмерного Массива:

- это объект, с которым может работать пользователь (таблицу значений можно вывести на экран, пользователь может ее заполнять, в дальнейшем введенные данные можно читать);
- построение индексов для быстрого поиска;
- клонирование, заполнение определенным значением всей колонки, выгрузка все колонки в массив.

Таблица Значений в Синтакс-помощнике представлена на рисунке ниже:

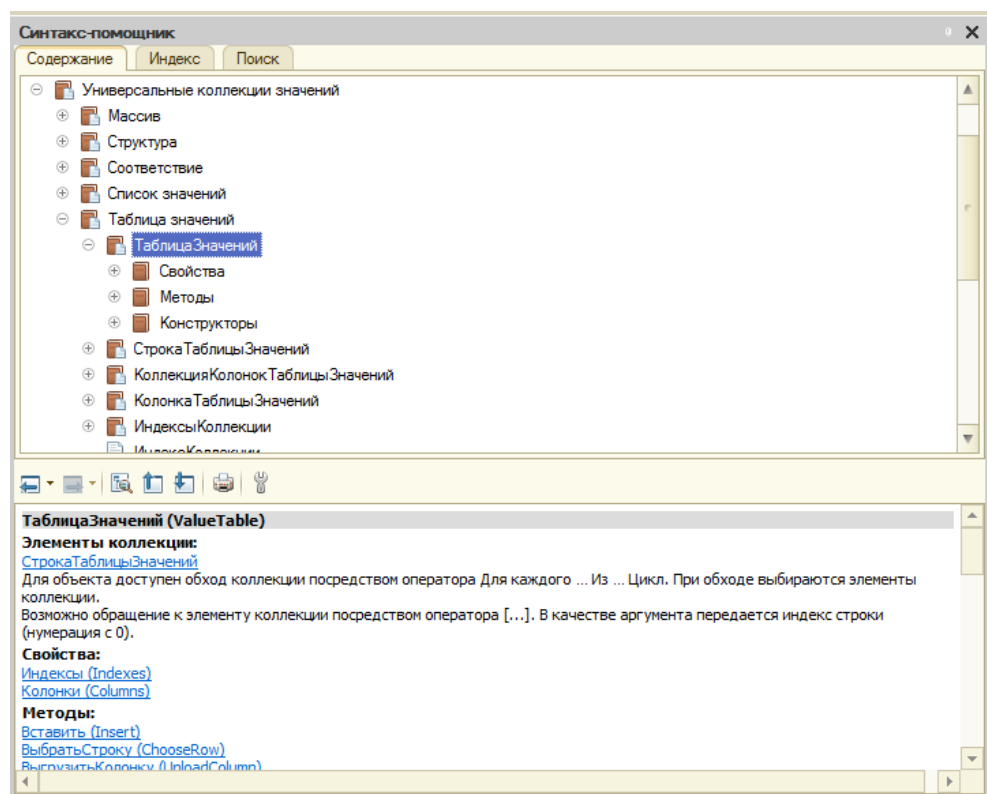


Таблица Значений используется как некий буфер хранения информации.



```

// Создание таблицы значений
ТаблицаЗначений = Новый ТаблицаЗначений;
// добавим в таблицу значений три колонки
ТаблицаЗначений.Колонки.Добавить ("Отдел", "Отдел");
ТаблицаЗначений.Колонки.Добавить ("Сотрудник", "фамилия сотрудника");
ТаблицаЗначений.Колонки.Добавить ("Оклад", "Оклад");

// добавим строку
Стр=ТаблицаЗначений.Добавить ();
Стр.Отдел="Отдел 1";
Стр.Сотрудник="Иванов";
Стр.Оклад=5600;

// добавим новую колонку
ТаблицаЗначений.Колонки.Добавить ("Стаж", "Стаж работы");
// ввод новой строки и данных по строке
ТекСтр = ТаблицаЗначений.Добавить ();
ТекСтр.Отдел = "Отдел 2"; ТекСтр.Сотрудник = "Петров";
ТекСтр.Оклад = 6700; ТекСтр.Стаж = 22;

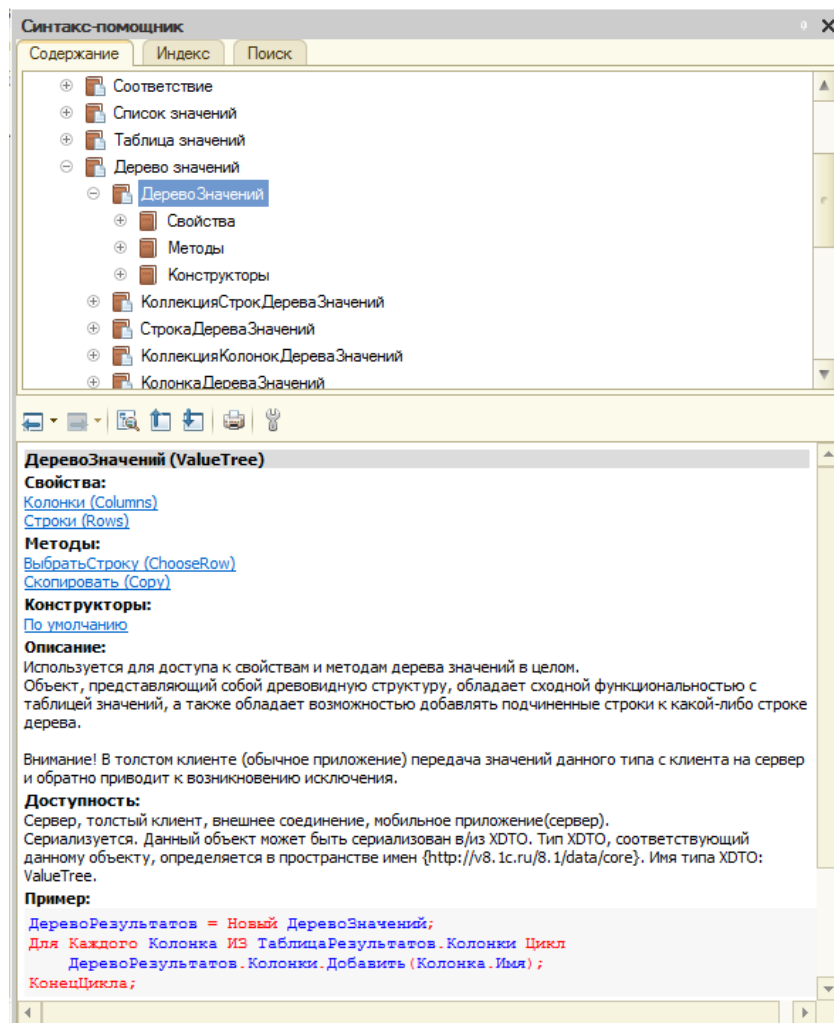
```

К Таблице Значений возможно построить запрос.

Дерево значений. Дерево значений представляет собой динамически формируемый набор значений любого типа, похожий на таблицу значений. В отличие от таблицы значений, строки дерева значений могут образовывать иерархические структуры: каждая строка дерева может иметь набор подчиненных строк, каждая из подчиненных строк, в свою очередь, также может иметь набор подчиненных строк и так далее. При этом поиск значений, сортировка, получение итогов могут осуществляться либо по текущему уровню иерархии, либо включая все подчиненные.

Дерево Значений тоже может быть отражено на экране.

Описание Дерева значений в Синтакс-помощнике представлено на рисунке.



COMSafeArray. Представляет собой объектную оболочку над многомерным массивом SAFEARRAY из COM. Позволяет создавать и использовать SAFEARRAY для обмена данными между COM-объектами.

Фиксированный массив. Неизменяемый массив. Массив заполняется системой при инициализации объектов данного типа или разработчиком, с помощью конструктора.

## 8. Конструкции языка

Конструкция языка 1с	Примечание
//Это комментарий	Так оформляются комментарии. Комментарий – это подсказки, пометки разработчика, которые помогают разобраться или вспомнить логику работы программного кода. Каждая новая строка комментария должна начинаться с символов //.
Перем ФИО;	Явное объявление переменной. <b>ФИО</b> – имя переменной.

A = 3;	Переменную можно не объявлять явно. При первом присвоении значения система создает данную переменную.
ЭтоЧисло = 23.5+12*2;	Переменной <b>ЭтоЧисло</b> присваивается числовое значение. С данными числового типа можно выполнять арифметические операции: сложение, вычитание, умножение и деление. <b>В качестве разделителя целой и дробной части используется точка!</b>
A = -0.123;	Числовые значения могут быть отрицательными.
A = (2+7) * 3; // в результате A=27	Приоритет выполнения арифметических операций: A = Сначала вычисляем 2 + 7, затем полученное значение умножаем на 3;
B = 2 * ((2+7) - 4); // в результате B = 10	B = Сначала вычисляем 2 + 7, затем от полученного значения отнимаем 4, затем полученное значение умножаем на 2.
ЭтоСтрока = "Пугачева";	Переменной <b>ЭтоСтрока</b> присваиваем строковое значение. <b>Значение строкового типа пишется в кавычках.</b>
ФИО = "Пугачёва" + " " + "Алла" + " "+"Борисовна"; // результат: ФИО = "Пугачёва Алла Борисовна"	Сложение строк. Символ " " прибавляем, чтобы между фамилией, именем и отчеством были пробелы. Вторая строчка кода это просто комментарий.
ДатаОтчета = '2013.01.01';	Переменная, которая хранит дату. <b>Значение типа Дата записывается в одинарных кавычках.</b>
ЧислоСекунд = '2013.01.02' - '2013.01.01'; // ЧислоСекунд = 86400	Даты можно вычитать одну из другой. В результате получим разницу между датами, измеренную в секундах. В сутках <b>86 400</b> секунд (60 сек * 60 мин * 24 ч).
НоваяДата = '2013.01.01'+86400; //НоваяДата = '2013.01.02'	К дате можно прибавлять и вычитать <b>число</b> . В результате к дате либо прибавится, либо отнимется число секунд.
Процедура РассчитатьЦену (Товар) КонецПроцедуры	Простая процедура. Между словами Процедура иКонецПроцедуры записывается текст процедуры.
Функция РассчитатьНалог(Сотрудник, НекаяДата) Возврат Налог; КонецФункции	Функция должна возвращать результат в место ее вызова.
Если Доход > 20000 Тогда Результат = "Жить можно"; Иначе Результат = "Так жить нельзя!"; КонецЕсли;	Простое условие. После слова КонецЕсли должна быть точка с запятой, потому что так заканчивается операторЕсли.

Результат = ?(Доход > 20000, "Жить можно", "Так жить нельзя!");	Сокращенное Если. Краткая запись предыдущего простого условия.
Если Доход > 20000 Тогда Результат = "Жить можно"; ИначеЕсли Доход > 10000 Тогда Результат = "Плохо"; Иначе Результат = "Так жить нельзя!"; КонецЕсли;	Множественное условие. Если первое условие не выполняется, то проверяется второе. Если ни одно из условий не выполняется то выполняется блок Иначе.
Если (Доход > 20000) И (КодКатегории = 2) Тогда КонецЕсли;	Составное логическое выражение.
Пока Номер <= 15 Цикл КонецЦикла;	Простой цикл Пока (с неизвестным числом повторений). После слова КонецЦикла должна быть точка с запятой, потому что так заканчивается оператор Пока.
Для Номер = 1 По 15 Цикл КонецЦикла;	Простой цикл Для (цикл с известным числом повторений).
Для каждого СтрокаТаблицы Из Таблицы Цикл КонецЦикла;	Еще одна разновидность цикла. Оператор цикла Для каждого предназначен для циклического обхода коллекций значений (табличных частей справочников, документов и т.д.). При каждой итерации цикла возвращается новый элемент коллекции. Обход осуществляется до тех пор, пока не будут перебраны все элементы коллекции.
Пока <условие> Цикл Если <условие> Тогда Продолжить; КонецЕсли; КонецЦикла;	Оператор Продолжить передает управление в начало цикла.
Пока <условие> Цикл Если <условие> Тогда Прервать; КонецЕсли; КонецЦикла;	Оператор Прервать производит досрочный выход из цикла. Управление передается на операторы после цикла.

## 9. Прикладные объекты «Обработки»

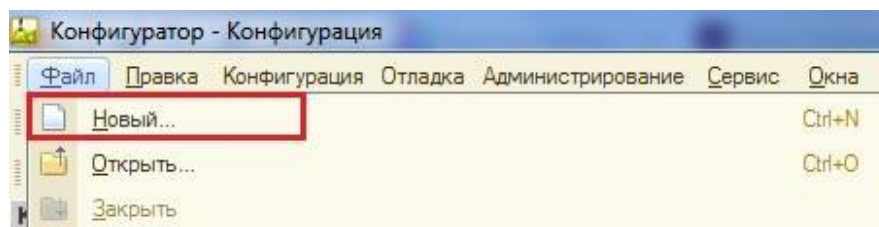
Обработки – объекты конфигурации служащий для изменения и преобразования информации в базе данных.

Обработки 1С подразделяются на **внутренние** и **внешние**.

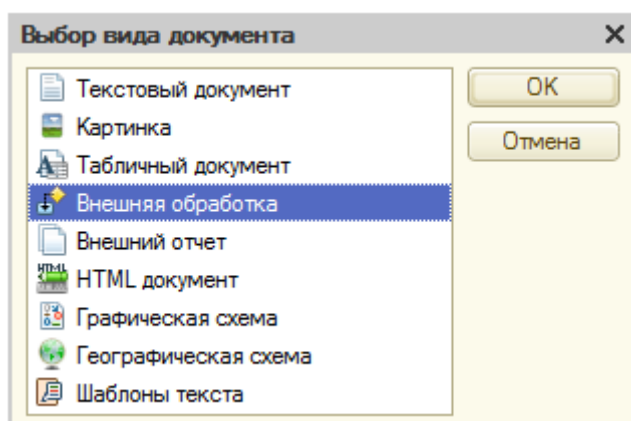
**Внешние обработки 1С** – это программы 1С, которые хранятся в отдельных файлах и могут передаваться на другой компьютер отдельно. При их изменении не нужно также сохранять конфигурацию заново. В Конфигураторе внешнюю обработку 1С можно открыть как файл и запустить в работу.

Создание внешней обработки:

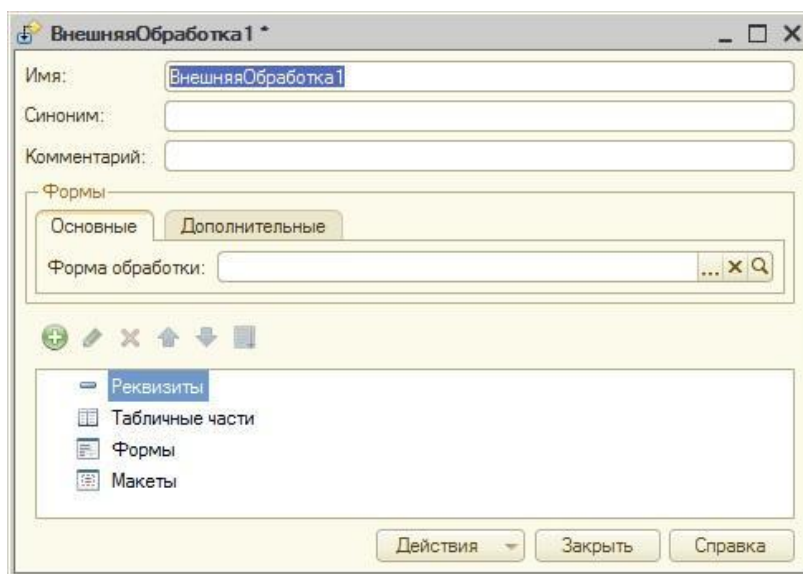
**Файл → Новый → Внешняя обработка.**



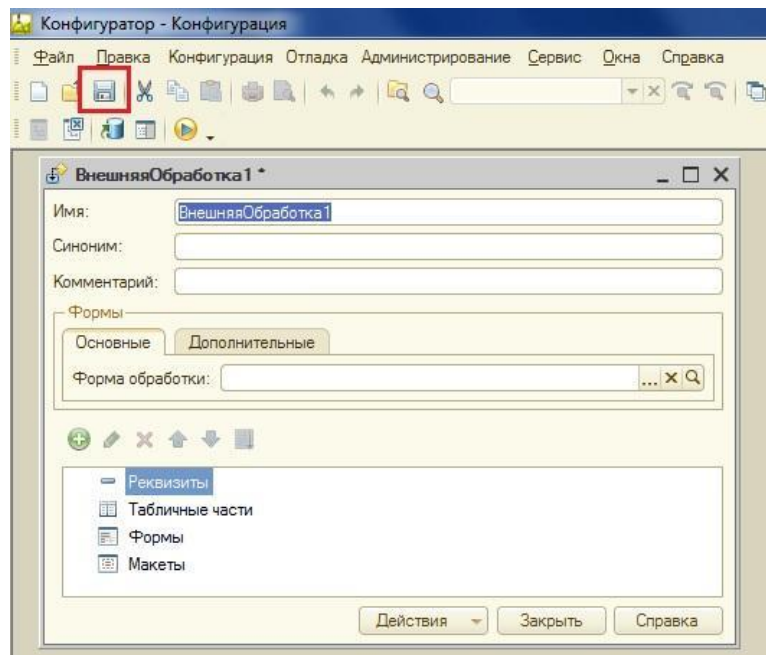
После откроется окно «Выбор вида документа», в котором необходимо выбрать «Внешняя обработка».



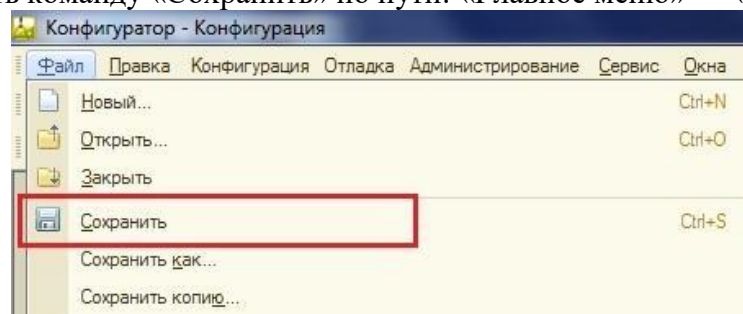
Откроется окно менеджера внешней обработки, и вы можете проделать с ней все нужные операции.



После того, как внешняя обработка создана, её желательно сохранить на жестком диске (и делать это периодически во время работы с ней в конфигураторе). Для того, чтобы сохранить внешнюю обработку необходимо или нажать на кнопку «Сохранить» меню «Стандартная».

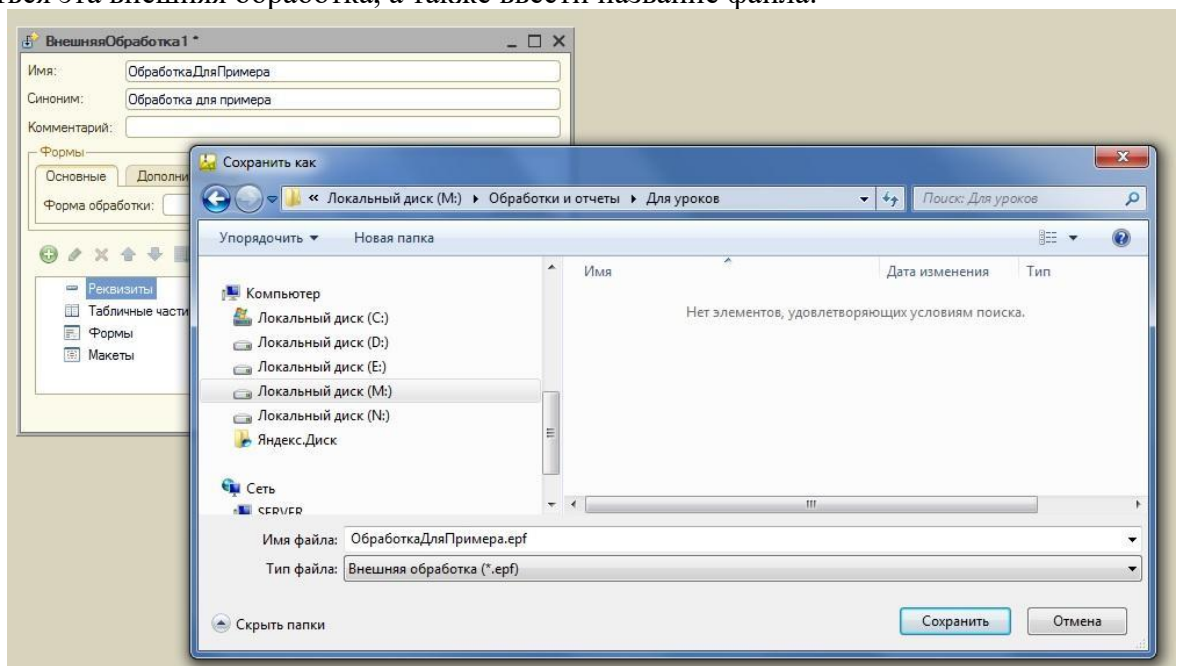


Или выполнить команду «Сохранить» по пути: «Главное меню» — «Файл».

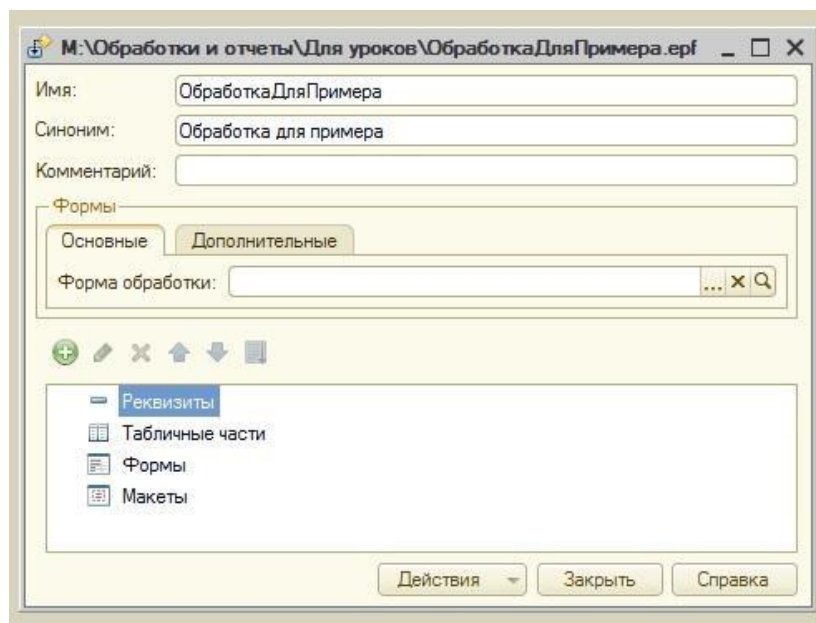


Или просто нажать сочетание клавиш Ctrl + S.

После этих действий откроется окно, в котором можно выбрать каталог, где будет храниться эта внешняя обработка, а также ввести название файла.

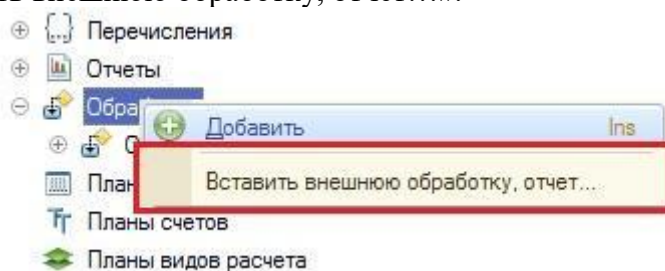


После сохранения, у внешней обработки сверху будет прописан путь к ней.

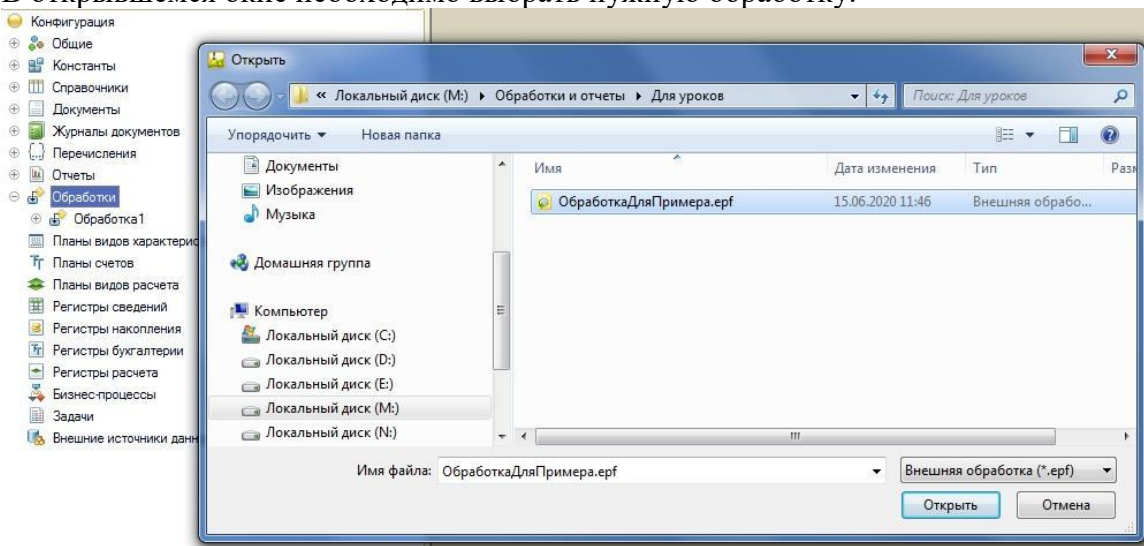


Файлы всех внешних обработок имеют расширение erpf.

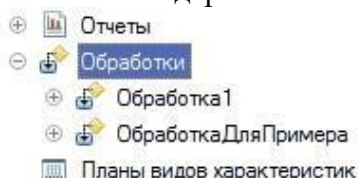
Добавить внешнюю обработку 1с. Иногда возникают задачи добавить внешнюю обработку в конфигурацию 1С. Чтобы это сделать, необходимо выделить ветку «Обработки» дерева метаданных конфигурации, вызвать контекстное меню и выполнить в нем команду «Вставить внешнюю обработку, отчет...».



В открывшемся окне необходимо выбрать нужную обработку.

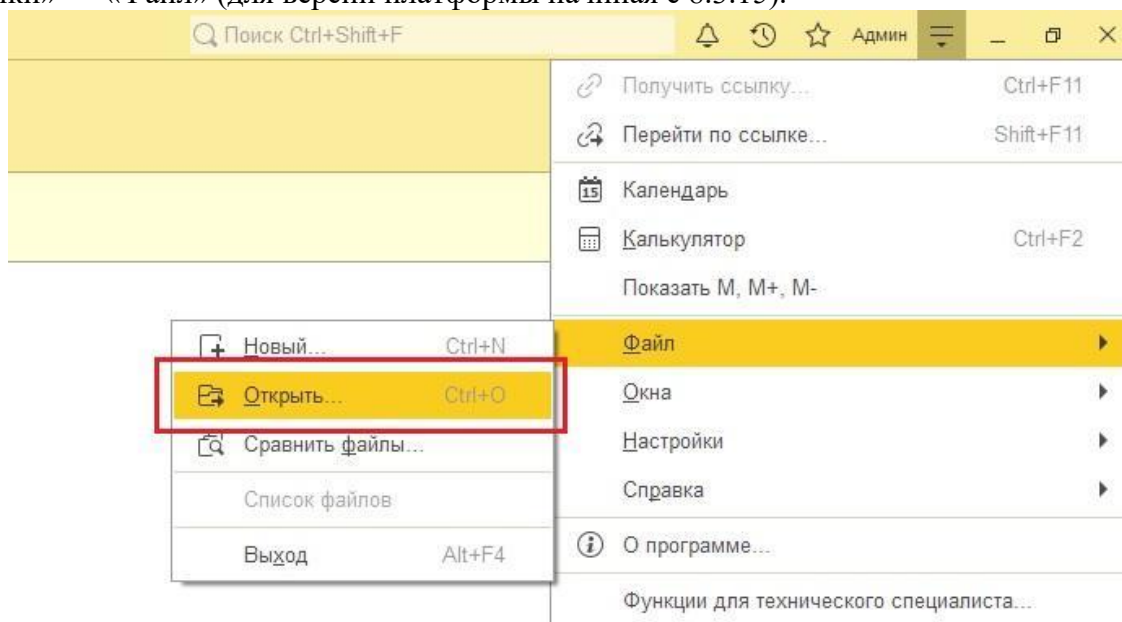


После, обработка появится в дереве.

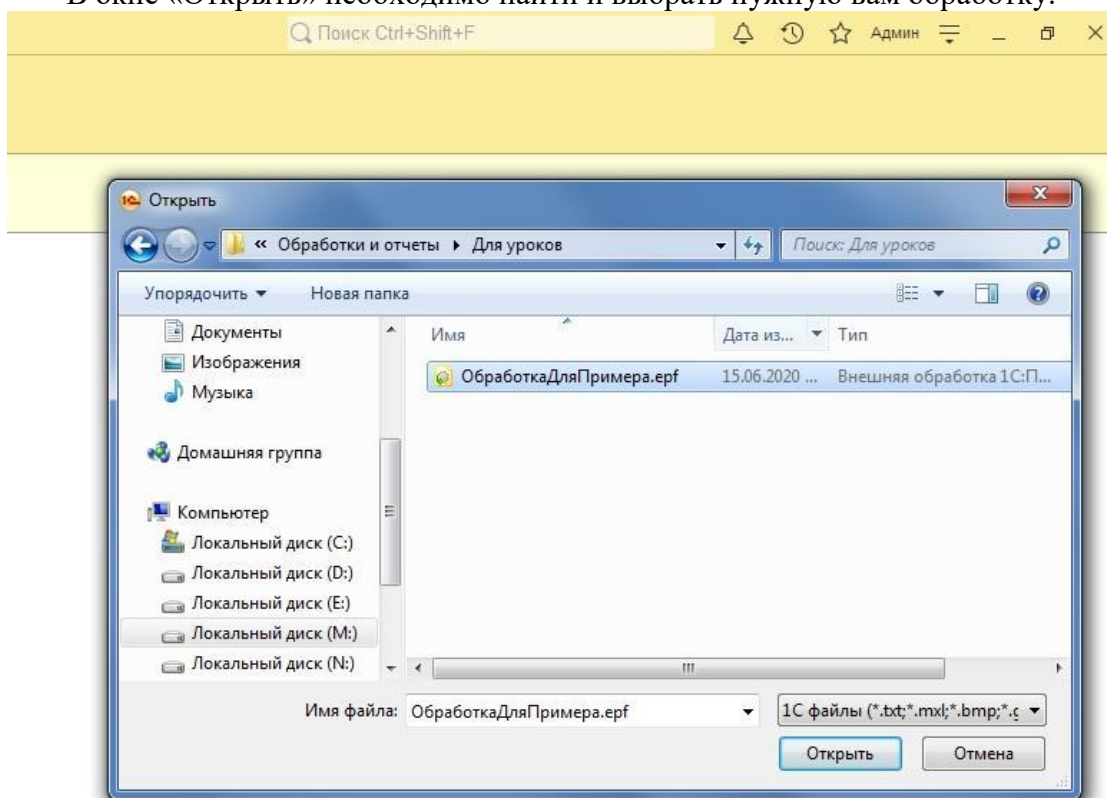




Открыть внешнюю обработку 1С. Если необходимо открыть внешнюю обработку, то следует выполнить команду «Открыть», которая находится по пути «Сервис и настройки» — «Файл» (для версии платформы начиная с 8.3.15).

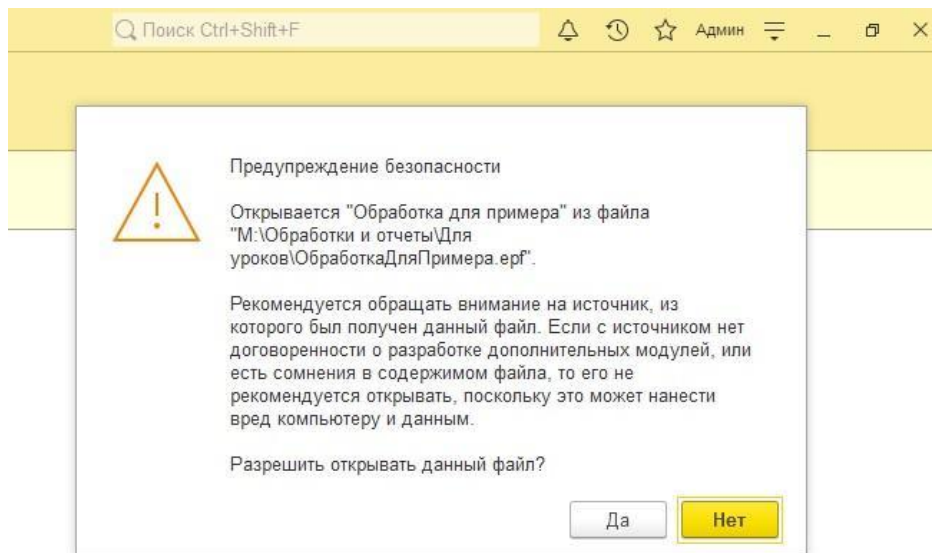


В окне «Открыть» необходимо найти и выбрать нужную вам обработку.

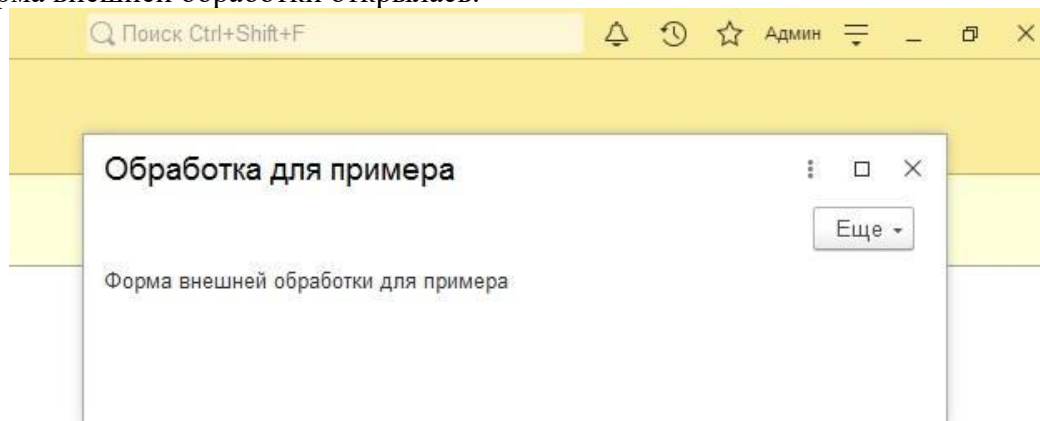


После выйдет предупреждение безопасности, если вы знаете, что это за обработка, то нажимаете кнопку «Да».

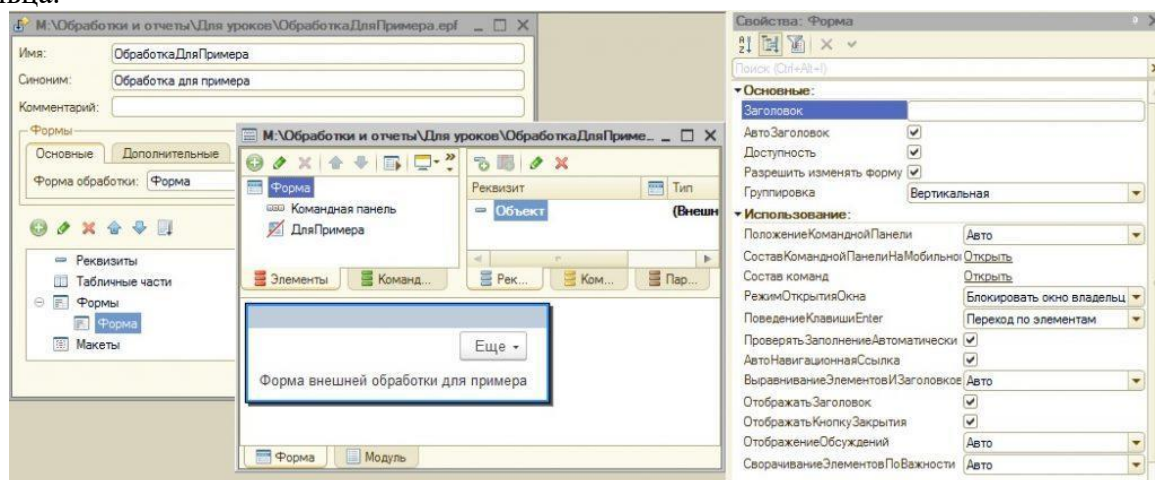




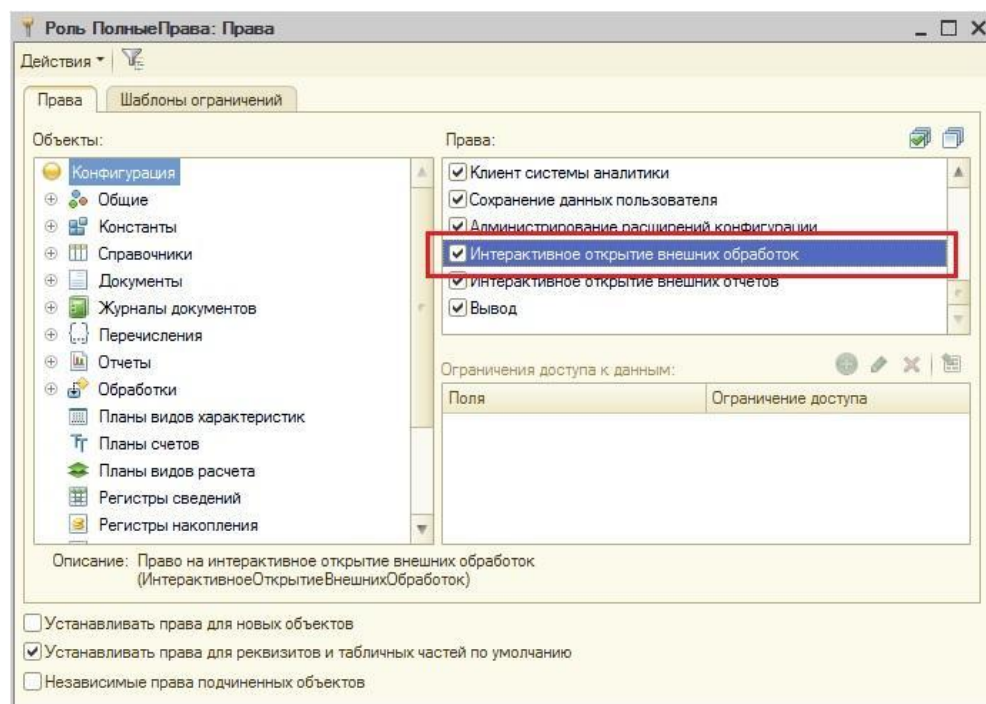
Форма внешней обработки открылась.



В данном случае открылась форма обработки в отдельном окне, потому что, во-первых, была создана основная форма внешней обработки, а во-вторых, установлена у основной формы в свойство «Режим открытия окна» значение блокировать окно владельца.



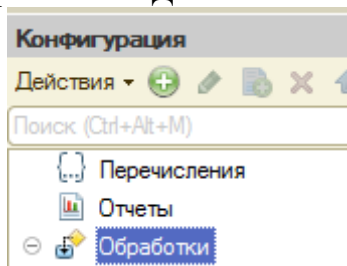
Чтобы пользователь мог открывать внешние обработки, ему необходимо установить право «Интерактивное открытие внешних обработок».



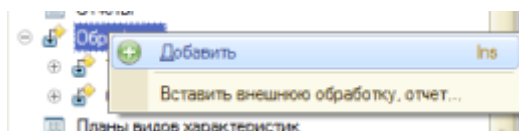
Внутренние или встроенные в Конфигурацию могут быть доступны пользователям программы 1С с помощью меню, кнопок, форм и других элементов интерфейса. Обычно они разрабатываются для конкретной конфигурации 1С, и помогают в работе при определенных ситуациях в работе пользователя.

Создание внутренней обработки:

Дерево конфигураций → Обработки → Добавить



**Пример 1.** Простейшие вычисления. Вычислить факториал числа.  
Создадим новую обработку.



Имя обработки – Факториал.

Обработка Факториал

Основные

Подсистемы

Функциональные опции

Данные

Формы

Команды

Макеты

Права

Прочее

Имя: Факториал

Синоним: Факториал

Комментарий:

Расширенное представление:

Пояснение:

Добавим новую Форму обработки.

Обработка Факториал

Основные

Подсистемы

Функциональные опции

Данные

Формы

Команды

Макеты

Права

Прочее

Формы:

Форма обработки: ... X Q

Добавить

Действия

<Назад

Далее>

Закреть

Справка

Конструктор формы обработки

Выберите тип формы:

☒ Форма обработки

☐ Произвольная форма

☒ Назначить форму основной

Имя: Форма

Синоним: Форма

Комментарий:

< Назад

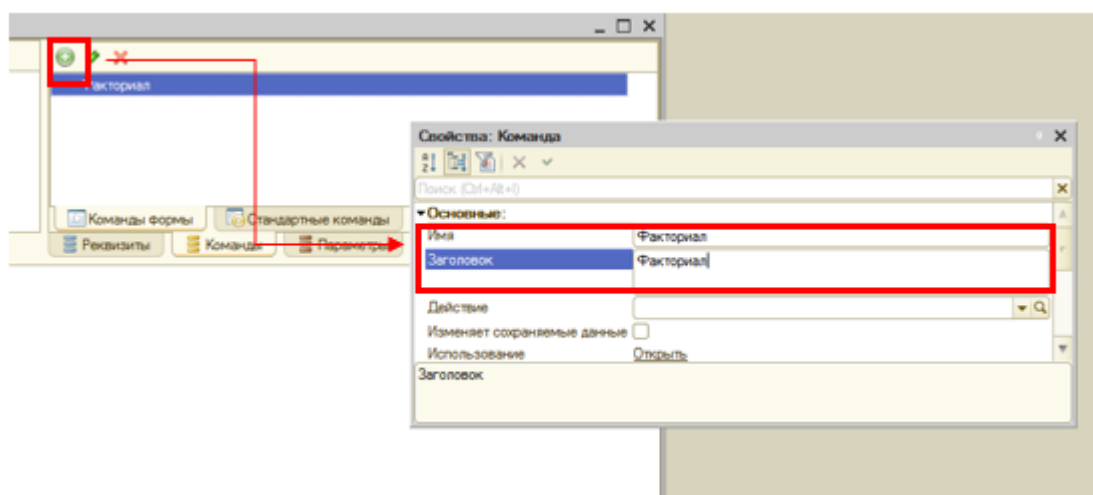
Далее >

Готово

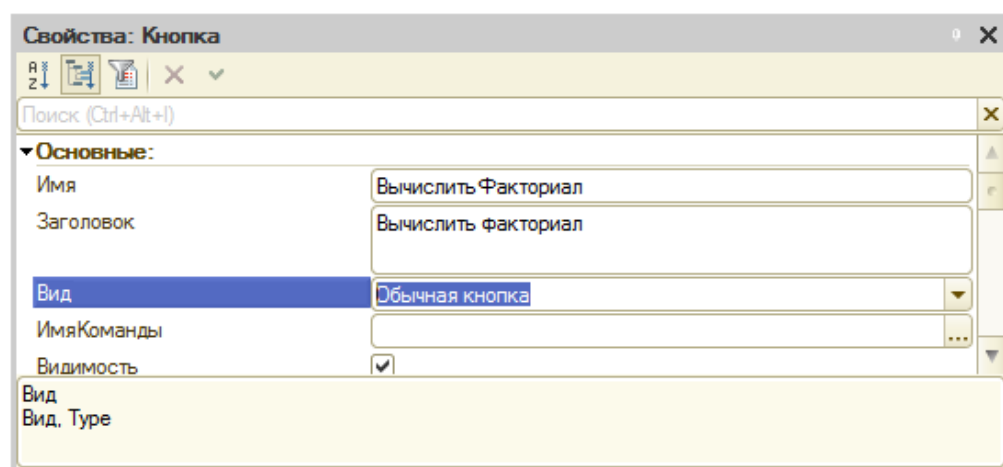
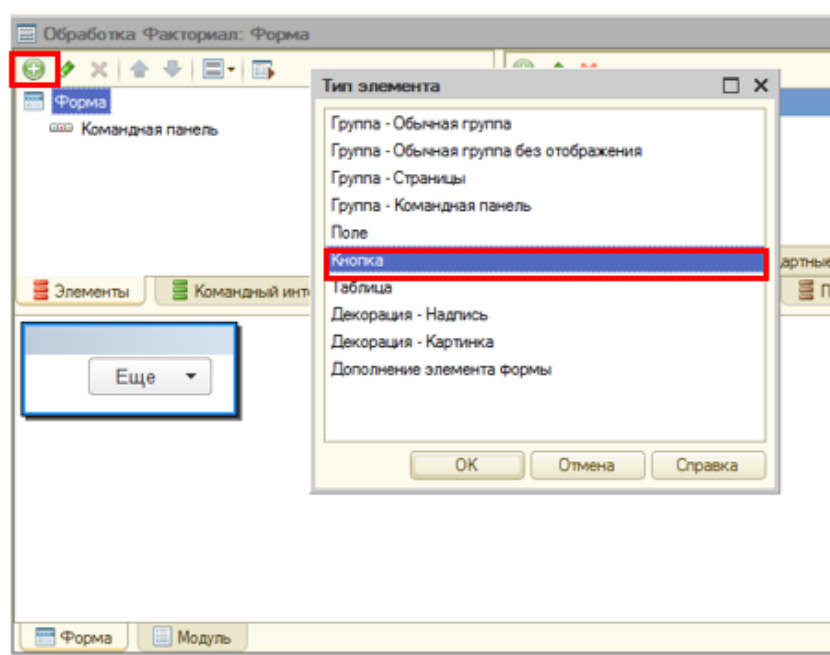
Отмена

Справка

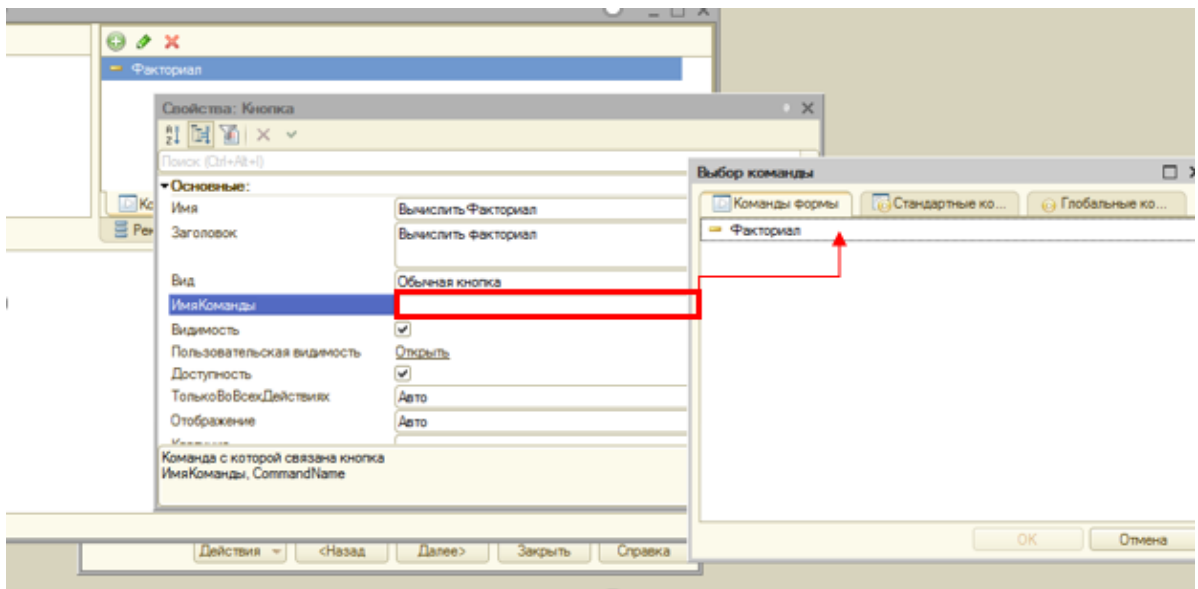
В форме обработки добавим **команду**, имя – Факториал.



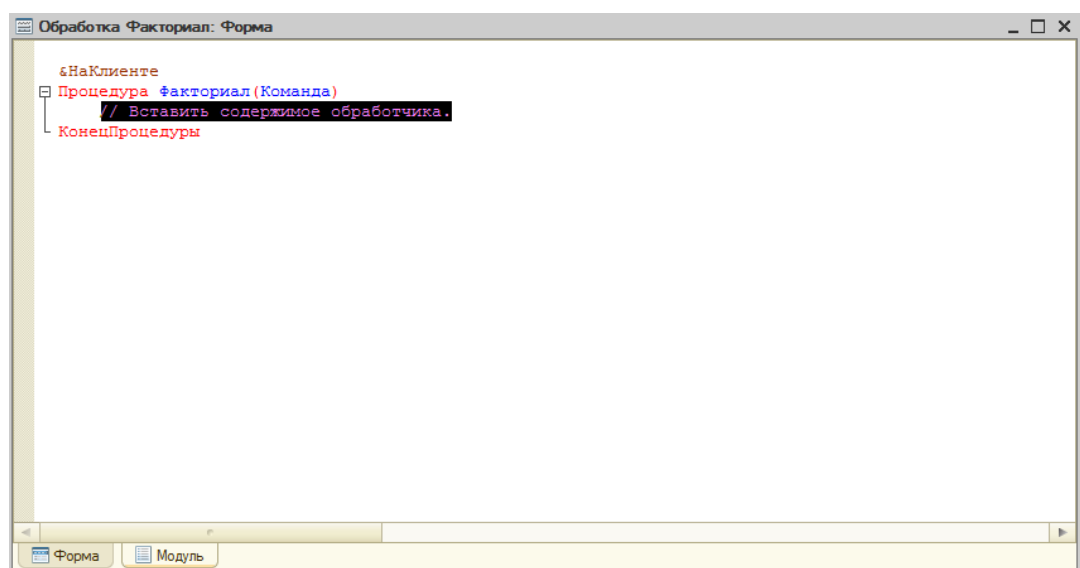
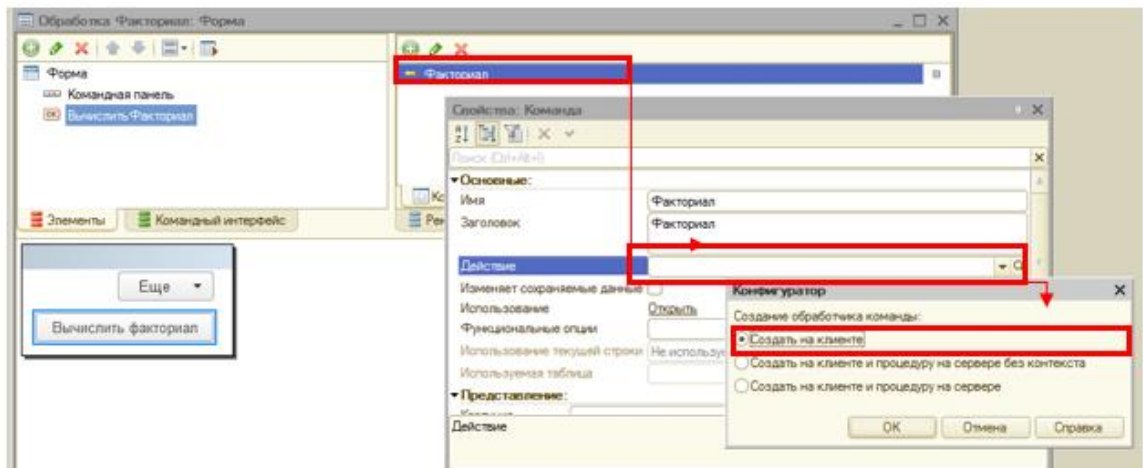
Добавим на форму новую кнопку Вычислить факториал.



Зададим имя созданной ранее команды Факториал.



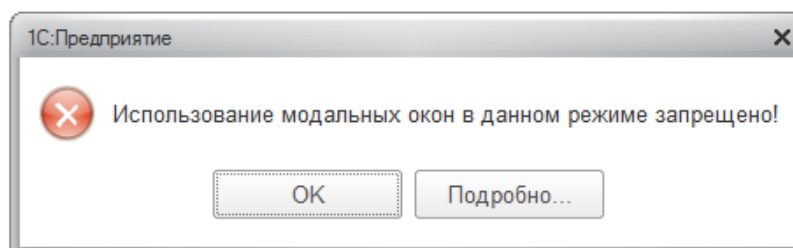
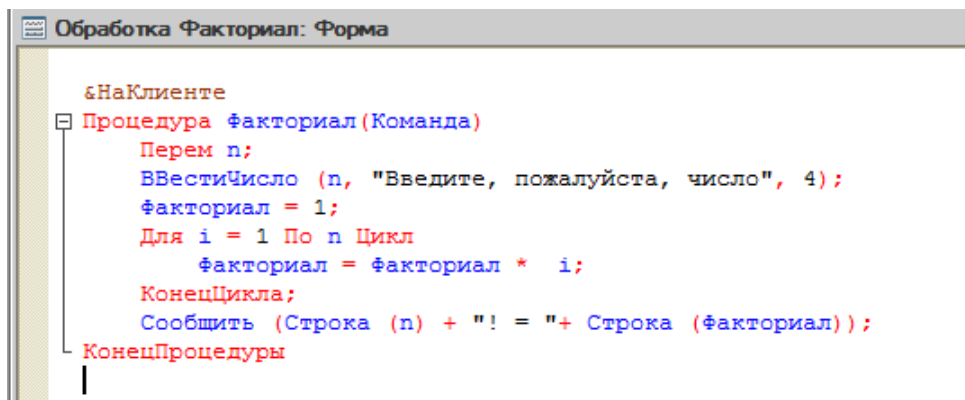
Необходимо задать действия для кнопки.



Обозначим переменные:

$n$  – число, вводимое пользователем;

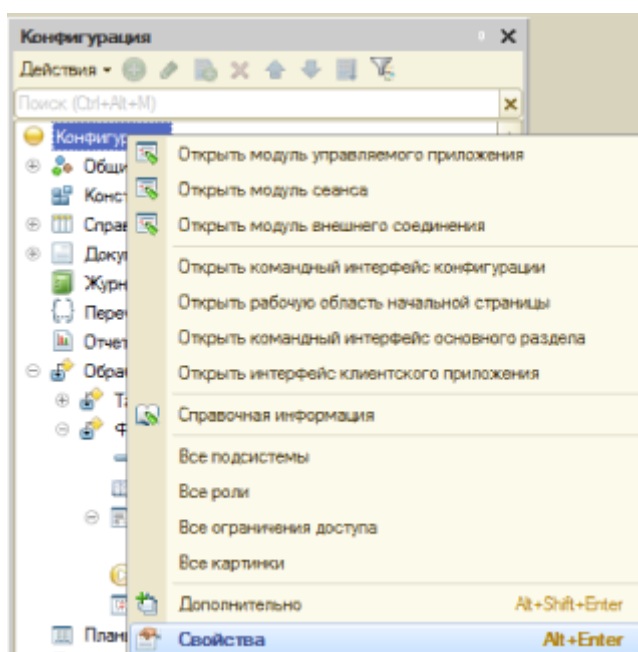
Факториал – факториал числа.

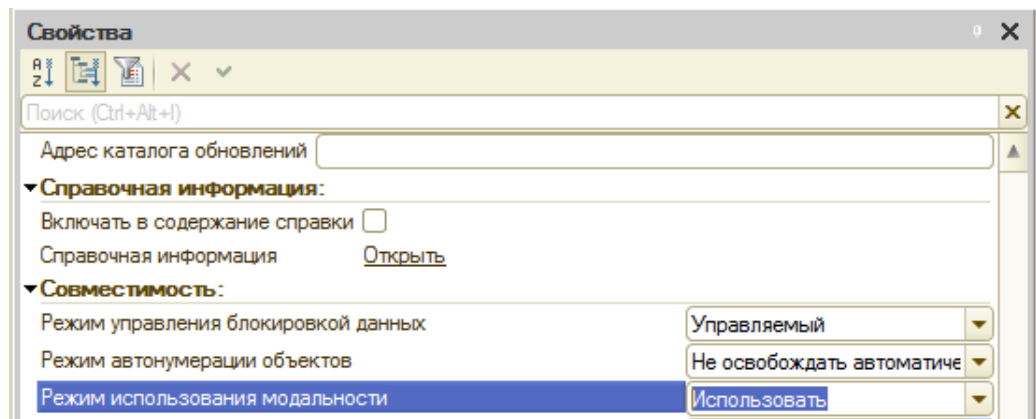


**Причина.** Разработчики технологической платформы 1С идут в ногу со временем, стандартизируя свои решения под мировые стандарты разработки программного обеспечения. Все стандарты, так или иначе, сводятся к единому интерфейсу, близкому к веб-страницам.

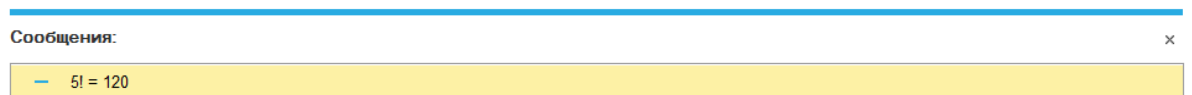
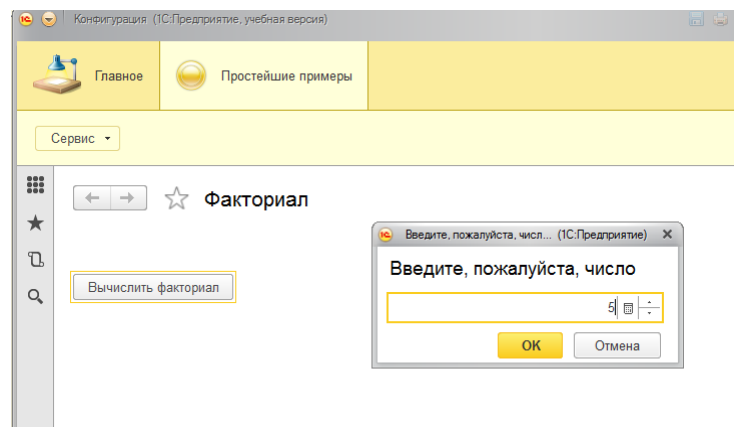
Модальные и всплывающие (pop-up) окна считаются плохим тоном и уже давно перестали быть нормальным при разработке софта.

**Решение проблемы (быстро, но неправильно):** в открытой конфигурации вызовите контекстное меню, выберите Свойства:





Результат изменений



**Решение проблемы (правильно).**

Правильный способ решения этой проблемы – доработать конфигурацию или внешнюю обработку под новые требования.

Встроенные операторы, которые вызывали модальные окна, необходимо заменить на дублирующие функции.

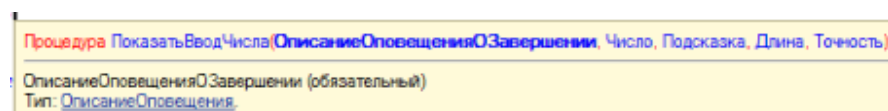
*Например:*

Предупреждение – ПоказатьПредупреждение

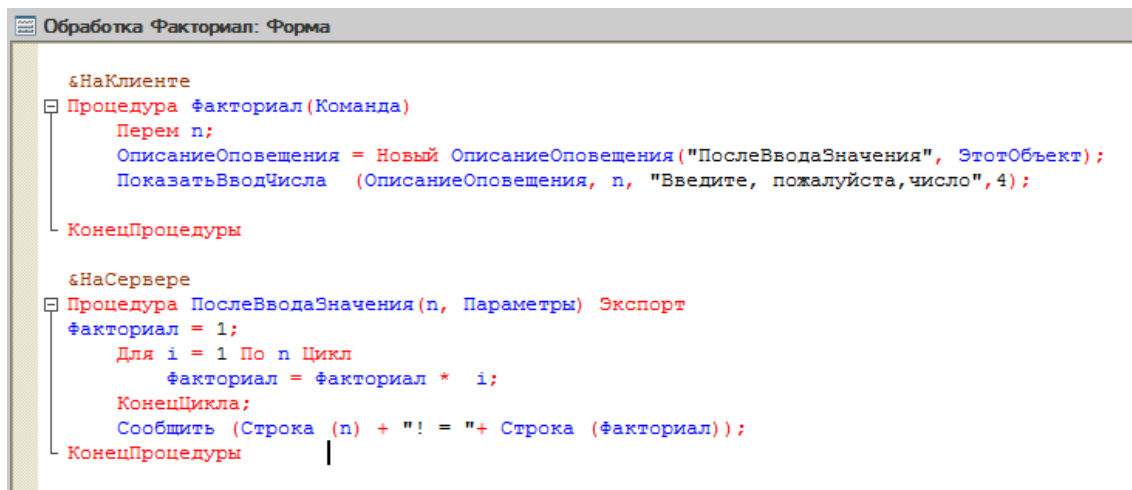
Вопрос – ПоказатьВопрос

**ВвестиЧисло – ПоказатьВводЧисла**

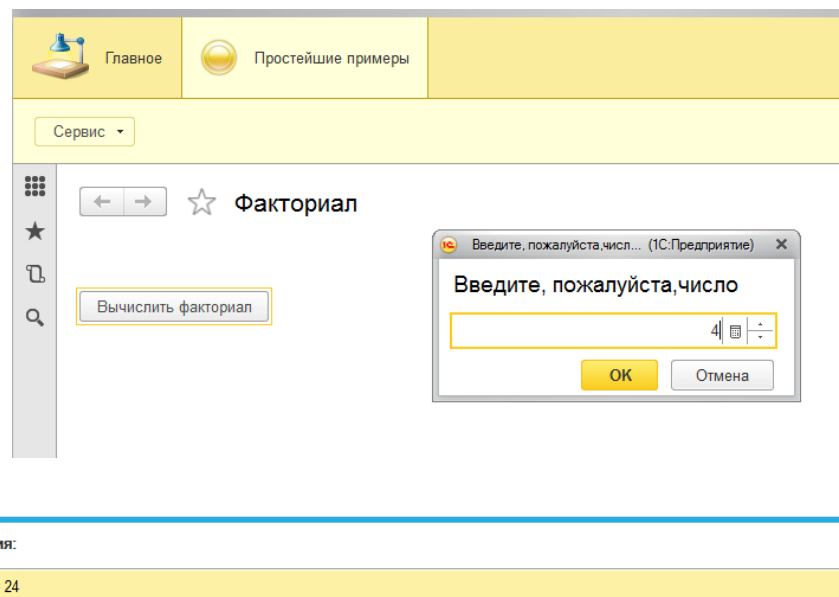
При этом появился специализированный объект – ОписаниеОповещения.



Внесите изменения.



Замените метод «ОткрытьФормуМодально» на «ОткрытьФорму».  
Результат изменений



## 10. Отладчик

Отладчик – вспомогательный инструмент, облегчающий разработку программных модулей системы 1С Предприятие.

Процесс отладки заключается в последовательном выполнении следующих действий:

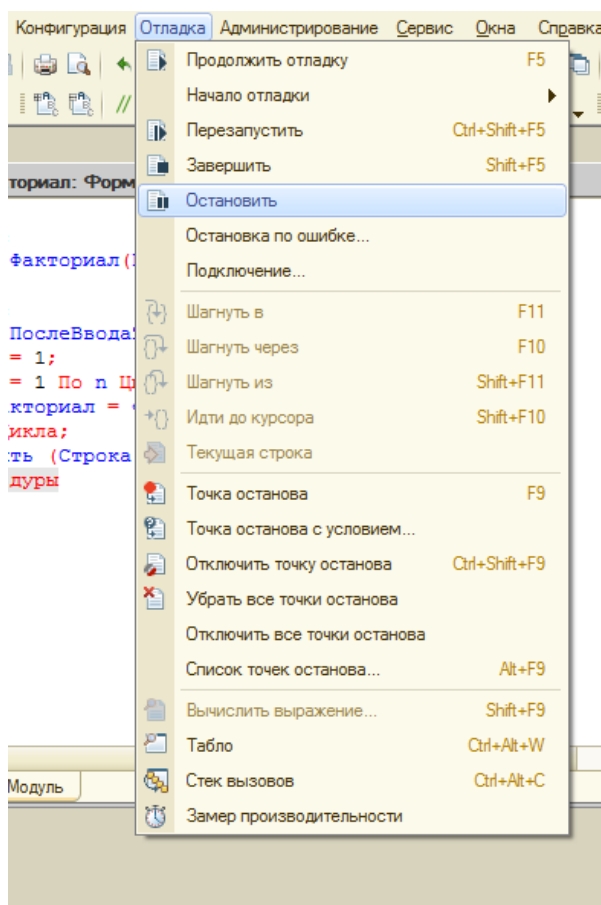
- Запустите конфигуратор и откройте модуль, предназначенный для отладки;
- Расставьте в требуемых строках модуля точки останова;
- Запустите режим «1СПредприятие» для выбранной конфигурации, если режим уже был запущен, то выполните команду «Отладка → Подключиться»;
- Выполните действия, которые вызовут исполнение отлаживаемого модуля;
- Проведите пошаговое выполнение нужного вам фрагмента модуля.

Для отладки модуля внешней обработки необходимо открыть файл внешней обработки в Конфигураторе, воспользовавшись пунктом «Файл → Открыть». В дальнейшем с модулем внешней обработки в Отладчике можно работать так же, как и с любым другим модулем. Отладчик позволяет установить на конкретную строку модуля специальный маркер – точку останова, – при достижении которой исполнение



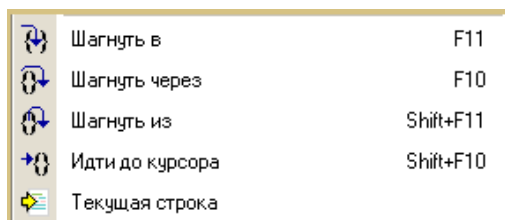
программного модуля останавливается и управление передается отладчику. Точку останова можно установить в любой строке модуля, в любой момент работы с Отладчиком.

Для управления точками останова используются команды меню «Отладка» или команды из контекстного меню, вызываемого из строки.



При большом количестве точек останова удобно использовать отдельное окно для работы с точками останова, позволяющее просматривать и редактировать их в едином списке.

После того, как при достижении точки останова управление прикладным решением передано отладчику, существует возможность дальнейшего исполнения модуля в нескольких режимах: пошаговое выполнения, исполнение вызова функции или процедуры, прерывание пошагового исполнения функции или процедуры, выполнения модуля до той строки, на которой стоит курсор или продолжение свободного выполнения модуля:



С помощью табло и диалога «Выражение» можно получить значения интересующих выражений. Стек вызовов позволяет проследить последовательность вызова процедур и функций.

	Вычислить выражение...	Shift+F9
	Табло	Ctrl+Alt+W
	Стек вызовов	Ctrl+Alt+C
	Замер производительности	

В случае, если требуется прервать процесс отладки в целом, снимите все точки останова со всех модулей и выполните команду "Отладка → Продолжить", если в данный момент сработала точка останова. Если необходимо прервать отладку только данного модуля, воспользуйтесь командой "Отладка → Прекратить".

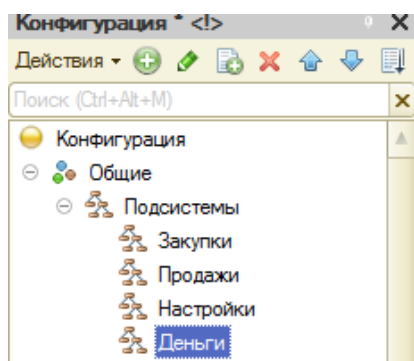
## 11. Пользовательский интерфейс

### 11.1 Создание Подсистем

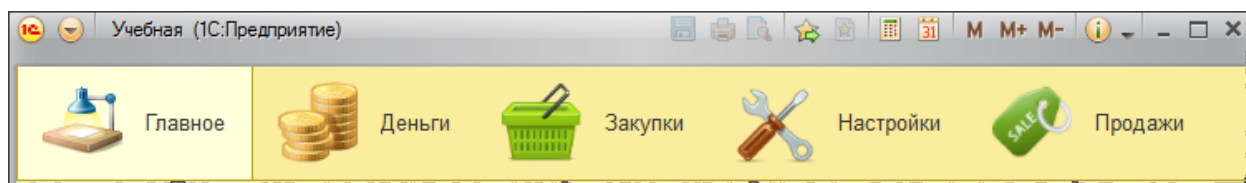
Подсистемы – это основные элементы для построения интерфейса «1С:Предприятия».

Объекты конфигурации Подсистема позволяют выделить в конфигурации функциональные части, на которые логически разбивается создаваемое прикладное решение.

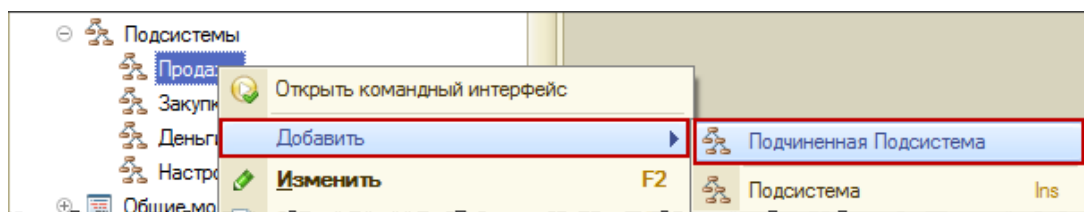
Эти объекты располагаются в ветке объектов Общие и позволяют строить древовидную структуру, состоящую из подсистем и подчиненных им подсистем (рис. 2.17).

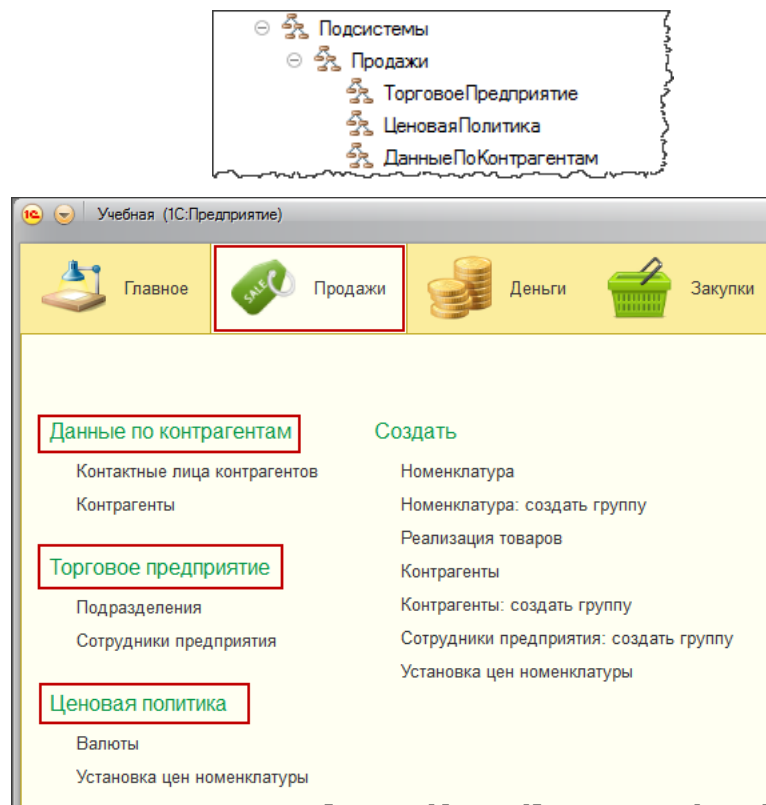


Подсистемы верхнего уровня являются основными элементами интерфейса, так как образуют разделы прикладного решения.



Для существующих Подсистем можно определять вложенные (подчиненные). Данные Подсистемы будут образовывать группы Панели навигации.



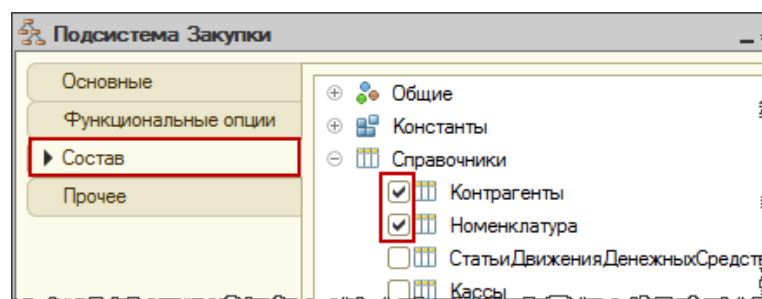


Не рекомендуется создавать подсистемы с уровнем вложенности больше двух, так как в противном случае интерфейс будет сложно читаемым.

В принципе, Подсистемы являются не обязательным объектом. Т.е. конфигурация вполне сможет работать без каких-либо Подсистем. Но в этом случае Панели разделов вообще не будет, все будет отображаться на Рабочем столе. Очень простые конфигурации с малым набором объектов смогут работать и без Подсистем. Но если в конфигурации достаточно много Документов, Справочников и Регистров, использование Подсистем существенно облегчает работу пользователя.

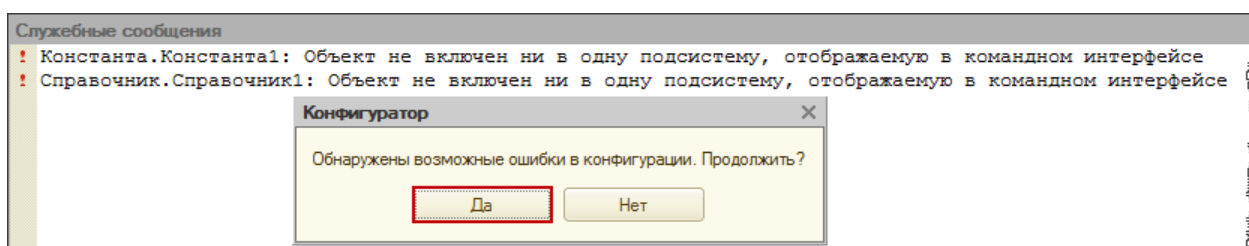
Каждый объект конфигурации (например, Справочник) может быть включен в одну или сразу несколько подсистем, в составе которых он будет отображаться.

Отметить, что некоторый объект конфигурации принадлежит какой-либо Подсистеме можно несколькими способами. Например, можно использовать «Окно редактирования» самой Подсистемы. На закладке «Состав» можно указать объекты, входящие в данную Подсистему.

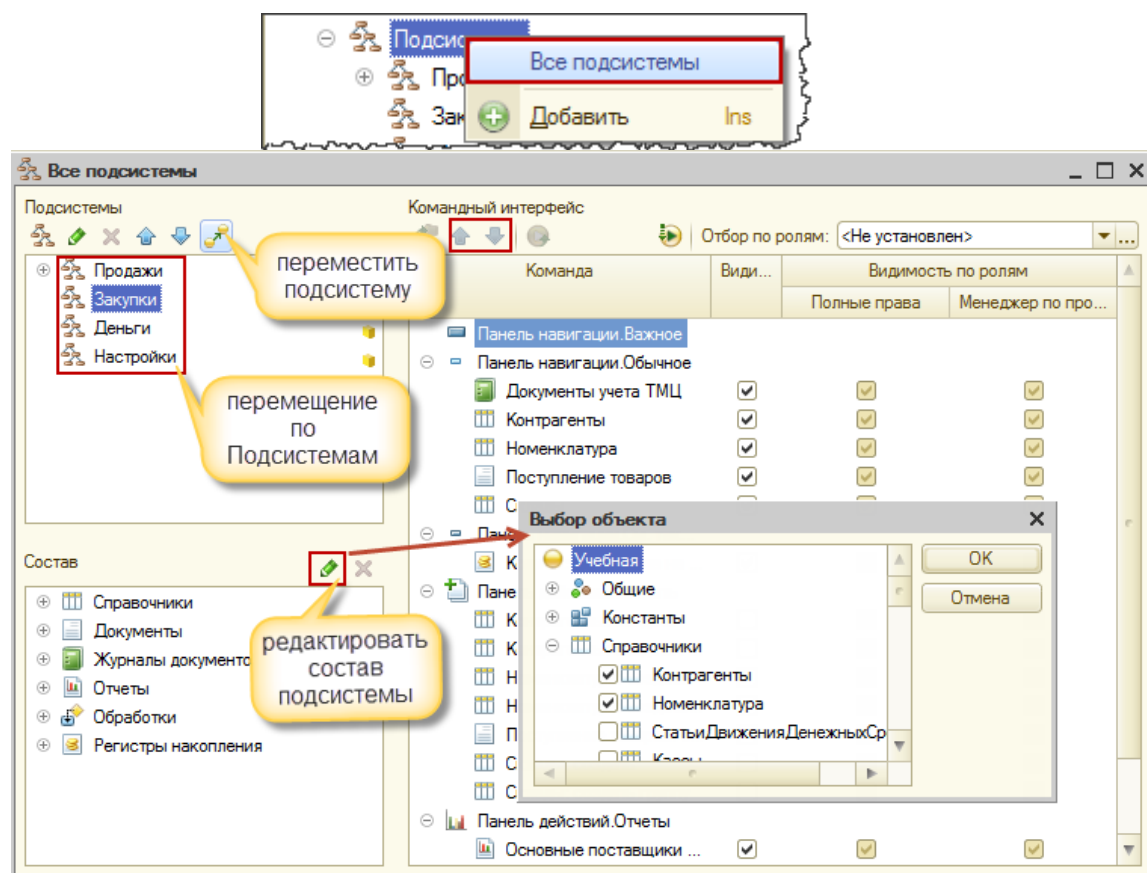


По умолчанию Платформа не проставляет привязку объекта ни к какой подсистеме. Т.е. разработчик должен сам зайти на эту закладку и проставить соответствующие галочки. Если разработчик этого не выполнит, то система определит отсутствие принадлежности к Подсистемам как ошибку. Но ошибка не является критичной, поэтому с этим можно согласиться. Фактически система сообщает о том, что, возможно, Вы забыли включить

новые объекты в Подсистемы. В этом случае объекты не будут отображаться в командном интерфейсе.



Иногда бывает необходимо сразу менять командный интерфейс в нескольких Подсистемах. В платформе 1С:Предприятие 8 существует сервисный инструмент, который позволяет редактировать командный интерфейс сразу нескольких Подсистем.



В открывшемся окне можно быстро перемещаться по Подсистемам и редактировать командные интерфейсы.

Таким образом, наличие подсистем определяет структуру прикладного решения, организует весь пользовательский интерфейс, позволяет рассортировать различные документы, справочники и отчеты по логически связанным с ними разделам, в которых пользователю будет проще их найти и удобнее с ними работать. При этом каждому конкретному пользователю будут видны лишь те разделы, то есть та функциональность прикладного решения, которые ему нужны в процессе работы.

С помощью подсистем, используя видимость по ролям, можно предоставить пользователю удобный и функциональный интерфейс, не содержащий лишних элементов (см. Приложение А).

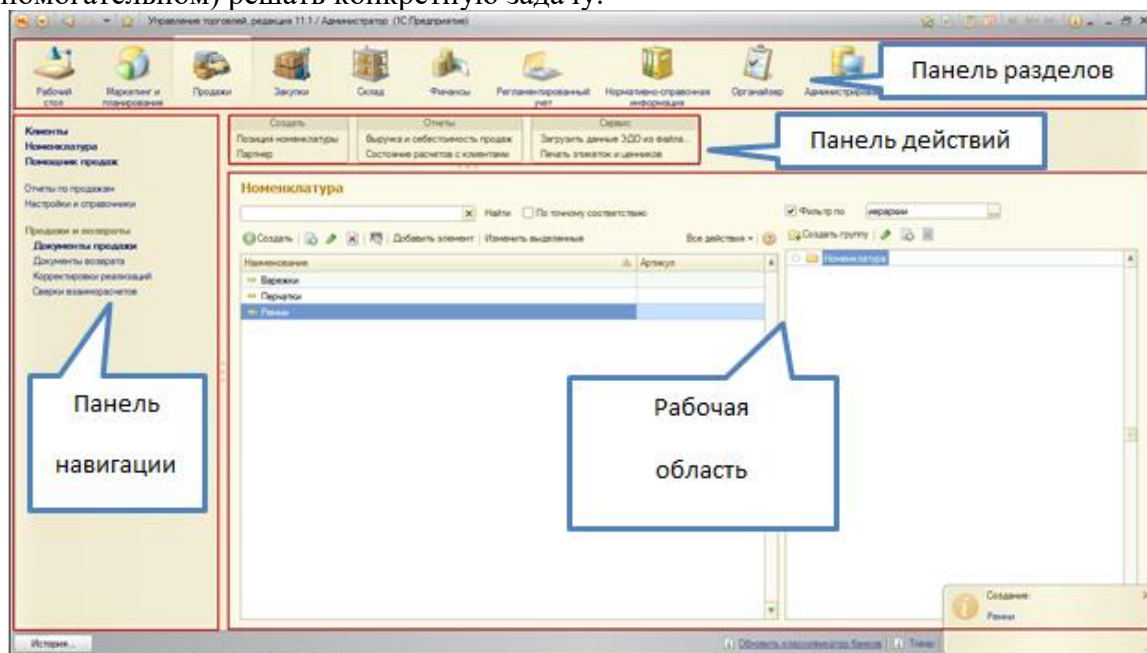
- Обычный интерфейс
- Управляемый интерфейс
- Интерфейс Такси

Возможность использования каждого вида интерфейса зависит от используемого клиента 1С: Предприятие 8. В Таблице 1 приведена совместимость клиентов 1С: Предприятие и видов пользовательских интерфейсов.

Вид клиента / Интерфейс	Обычный интерфейс	Управляемый интерфейс	Такси
Обычное приложение	Да	Да	
Тонкий клиент		Да	Да
WEB-клиент		Да	Да

Интерфейсы, релизованные в платформе «1С:Предприятие» отвечают современным требованиям современных. Пользовательский интерфейс «1С:Предприятие» называется «управляемым» потому, что с помощью прав и ролей пользователей, функциональных опций и настроек пользователя платформа автоматически строит командный интерфейс, управляемые формы и отчеты.

Пользовательский интерфейс «1С:Предприятие» можно охарактеризовать как задаче-ориентированный, позволяющий в каждом отдельном окне приложения (основном или вспомогательном) решать конкретную задачу.



*Основное окно* приложения предназначено для навигации по прикладному решению и вызова различных команд.

*Вспомогательное окно* предназначено для работы с объектами информационной базы (например, с документами или элементами справочников), построения отчетов или выполнения обработок данных.

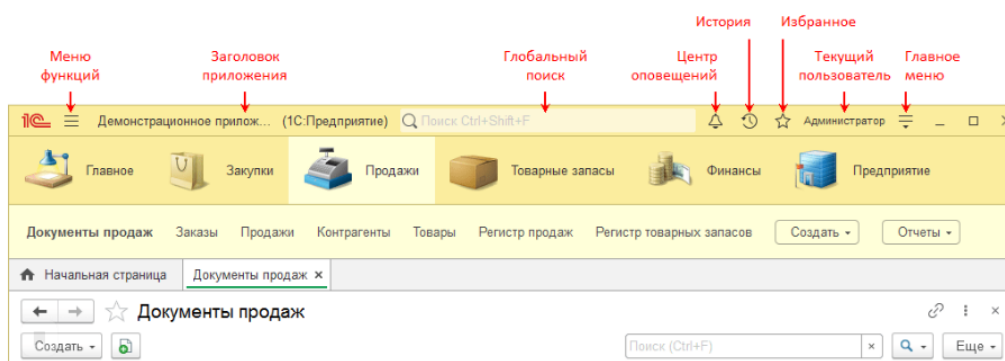
*Рабочий стол* – это стандартный раздел программы, содержащий часто используемые документы, отчеты, справочники и т. п. Этот раздел автоматически активизируется при запуске прикладного решения.

*Панель разделов* – это наиболее крупное деление функциональности прикладного решения. Она расположена в верхней части основного окна и соответствует верхнему уровню подсистем, добавленных в конфигурацию. С ее помощью осуществляется переход к другим разделам программы.

*Панель навигации* содержит своеобразное «оглавление» раздела. Она включает навигационные команды, которые позволяют перейти к той или иной точке этого раздела.

*Панель действий* содержит наиболее востребованные в повседневной работе команды, позволяющие быстро создавать новые объекты, выполнять типовые обработки или строить популярные отчеты.

Основное окно приложения также включает в себя главное меню системы и набор вспомогательных команд (калькулятор, календарь и пр.). А также предоставляет различные сервисные возможности, такие как: просмотр истории работы пользователя и возможность перемещаться по разделам прикладного решения, с которыми уже работал пользователь; добавление разделов, списков, объектов базы данных, отчетов и обработок в избранное; отображение в информационной панели (внизу окна) оповещений о последних действиях пользователя; возможность получить ссылку на любые разделы, списки, объекты базы данных, отчеты и обработки и перейти по полученной ссылке к этим данным и т. д.



Управляемый интерфейс может отображаться в *двух вариантах* внешнего вида:

- формы в **закладках**. При этом все окна будут отображаться в отдельных закладках одного родительского окна. Это больше похоже на классический интерфейс.
- формы в **отдельных окнах**. В этом режиме в основном окне отображается только одна форма. Новые формы могут отображаться в других окнах (переход между ними будет доступен за счет средств ОС). Это менее привычный вид приложения, но он имеет свои преимущества.

Кроме уже ставших привычными вариантов интерфейса в платформе 1С:Предприятие 8.3 появился интерфейс – **Такси**.

### 11.3 Интерфейс «Такси»

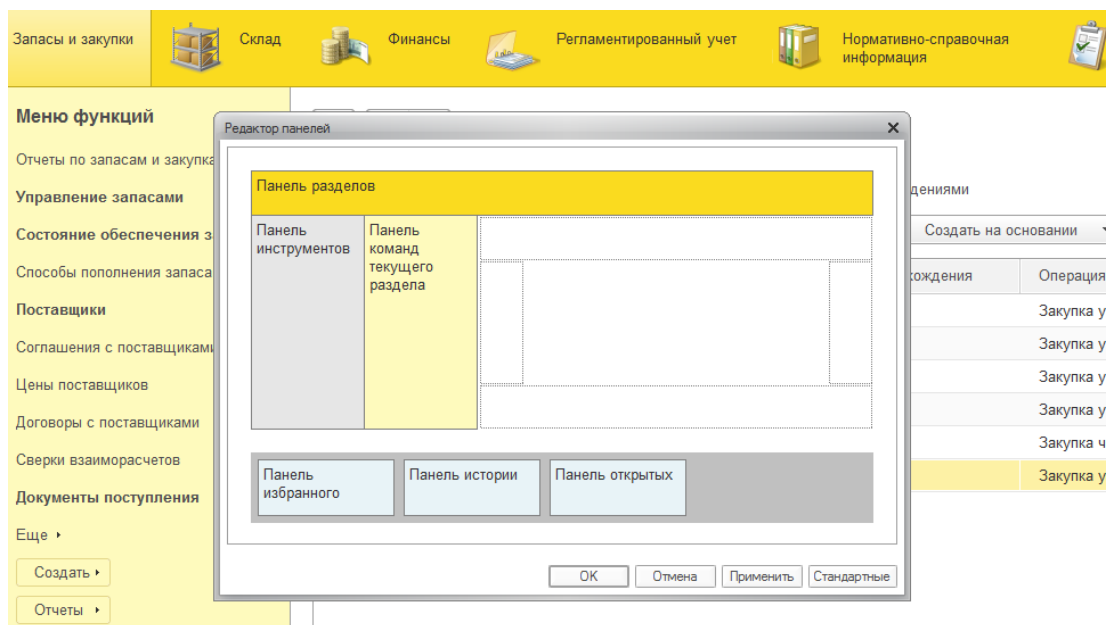
Интерфейс Такси – это новое слово в развитии технологической платформы 1С:Предприятие 8.3. Новое лицо программы адаптировано под стандарты веб-приложений.

При создании нового пользовательского интерфейса разработчики платформы ставили перед собой ряд задач. Поскольку с конфигурациями теперь возможно работать через интернет при помощи обычного веб-браузера, заметна ориентация платформы на работу с мобильными устройствами, на ввод информации при помощи сенсорного экрана, реагирующего на прикосновения к нему.

Новый интерфейс имеет *отличительные визуальные особенности*:

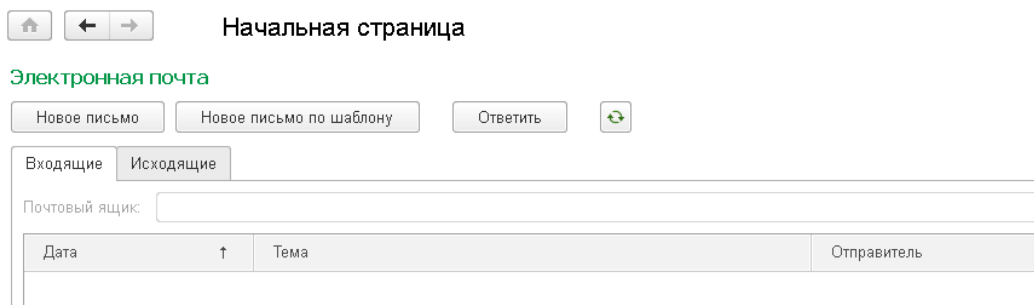
- максимизация рабочего пространства на мониторах с различным разрешением;
- современный дизайн;
- крупный шрифт;
- удобная навигация за счет использования вспомогательных панелей: "Инструменты", "Избранное", "История" и "Полнотекстовый поиск";
- понятный и удобный поиск в списках, управление поиском;
- возможность самостоятельно конструировать рабочее пространство (методом drag&drop).





- адаптация под стандарты веб-приложений;
- расширенные возможности выбора в полях ввода.

При использовании интерфейса Такси приложение представляет из себя основное окно, внутри рабочей области которого открываются все остальные окна. Окна не отображаются отдельно на панели задач Windows, даже если визуально рисуются поверх основного окна (например, формы выбора элемента справочника). Работа начинается с начальной страницы. Эту страницу невозможно закрыть.



Предусмотрена возможность настраивать в пользовательском режиме формы, отображаемые на начальной странице. Для этого в главном меню следует выбрать пункт Вид → Настройка начальной страницы.

Основное окно приложения состоит из нескольких панелей и областей, каждая из которых имеет свое функциональное назначение. При помощи клавиши F6 можно переключать фокус между формами, расположенными на начальной странице.

**Панель разделов.** Панель разделов отображает список подсистем конфигурации верхнего уровня. Первым слева всегда отображается раздел с заголовком Главное.

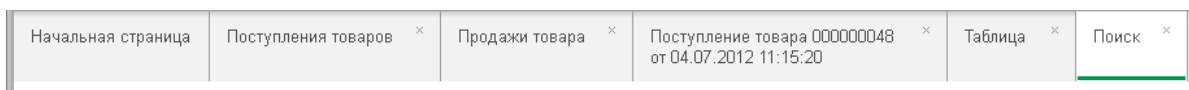


При щелчке мышью на конкретном разделе в панели функций текущего раздела представлены команды выбранного раздела.

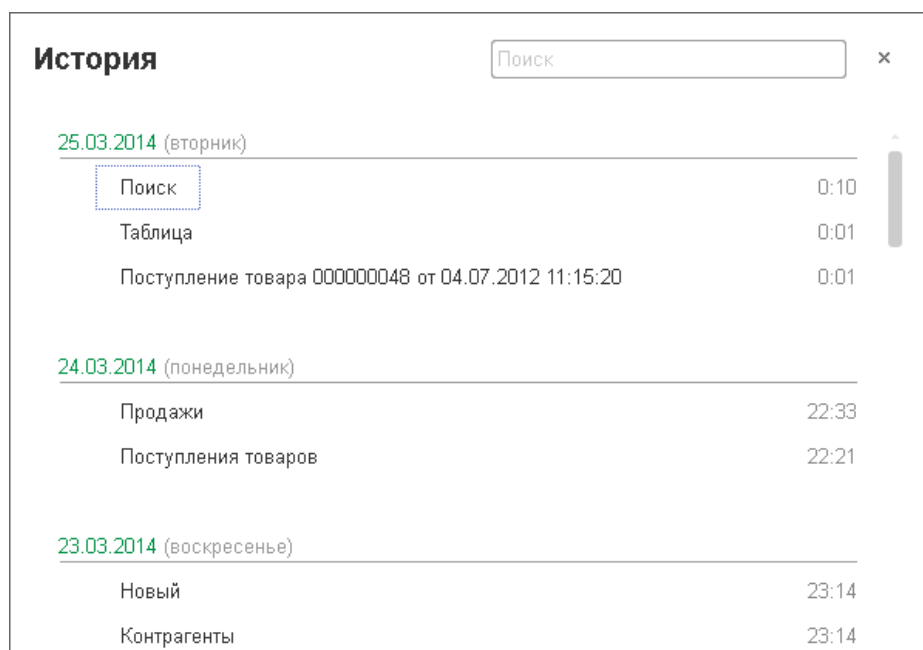




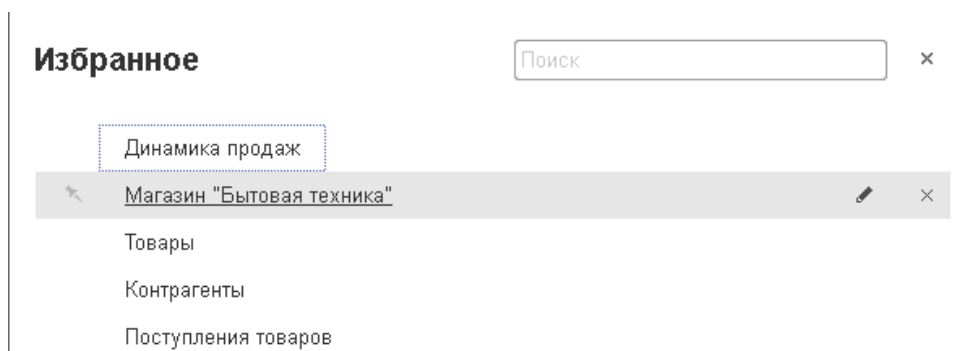
**Панель открытых.** На этой панели выводится список открытых в настоящий момент окон. Формы рабочей области начальной страницы не отображаются как отдельные формы в панели открытых.





**Панель истории.** Панель истории ускоряет доступ к использованным в последнее время объектам – открытым, отредактированным или добавленным справочникам, документам и т.д. Записи разделены по дням, а напротив каждой строчки указано время использования. В пределах одного дня записи упорядочены по времени доступа.



**Панель избранного.** В избранном хранятся данные и команды.



Добавить объект в избранное можно при помощи щелчка мышью на значке с изображением звездочки.

Дата	↓	Номер	Покупатель	Склад
	07.08.2012 10:15:00	000000007	Шлюзовая ООО	Малый
	10.08.2012 12:35:12	000000008	Магазин "Продукты"	Малый

На панели избранного также присутствует **строка поиска**. Заголовки элементов, расположенных в избранном, можно редактировать. Для этого предназначена небольшая кнопка с карандашом в правой части строки избранного. В платформе реализована специальная форма полнотекстового поиска. Эта форма доступна только в том случае, если полнотекстовый поиск включен.




**Поиск**

Ботинки

1. [Товар: Ботинки](#)  
Наименование: Ботинки
2. [Продажа: Продажа 000000004 от 04.08.2012 19:17:50](#)  
Товар: Ботинки
3. [Поступление товара: Поступление товара 000000021 от 07.07.2012 9:35:25](#)  
Товар: Ботинки

Вызвать форму поиска с клавиатуры можно при помощи сочетания клавиш **Ctrl + Shift + F**.

В нижней части основного окна приложения может существовать **информационная панель**. Она предназначена для отображения показателей производительности и индикации того, что включён режим имитации задержек при вызовах сервера. Показатели производительности отображаются в виде отдельной панели основного окна приложения.


(C) Текущие вызовы: 2 Накопленные вызовы: 328

Панель отображается, если в параметрах системы установлен флаг «Отображать показатели производительности»:

Параметры

☒ Отладка в текущем сеансе разрешена

☐ Устанавливать режим разрешения отладки при запуске

☒ **Отображать показатели производительности**

☐ Имитировать задержку при вызовах сервера:

Задержка при вызове (с.):

1,45

Задержка при передаче данных (с./Кбайт):

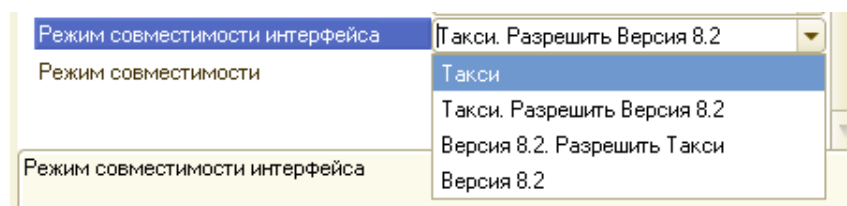
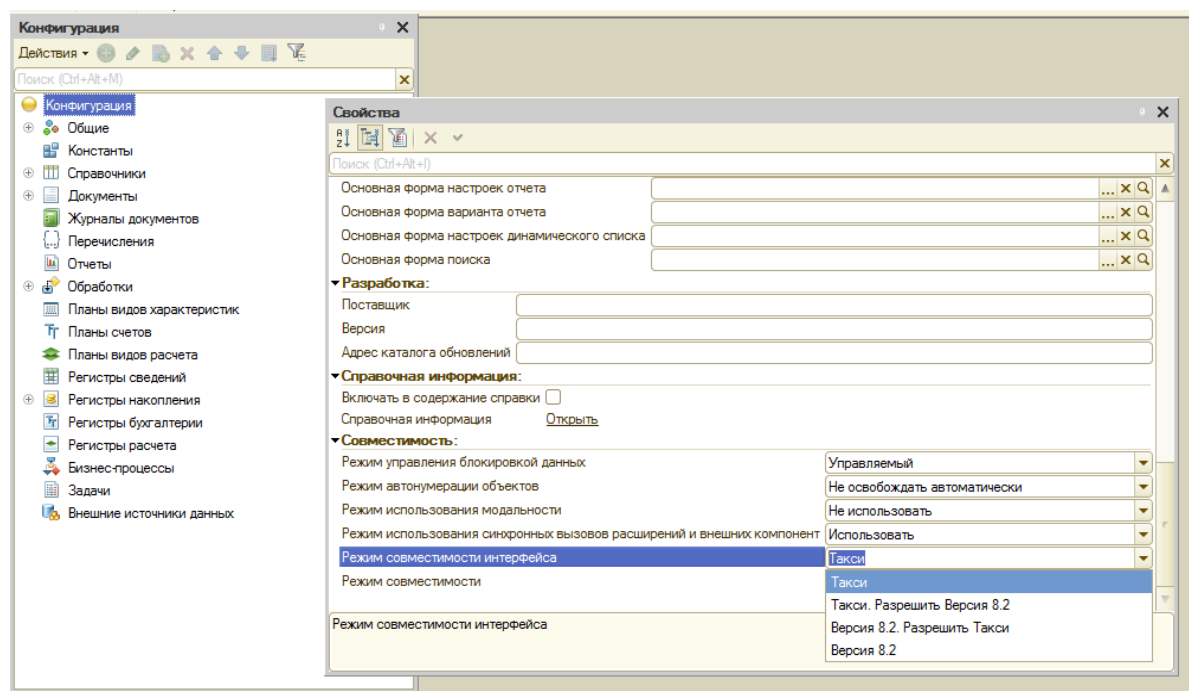
0,45

Задержка при получении данных (с./Кбайт):

0,15

☒ Отображать команду «Все функции»

Для того, что бы **включить** интерфейс Такси достаточно установить в свойствах конфигурации значение «Режим совместимости интерфейса» режим «Такси».



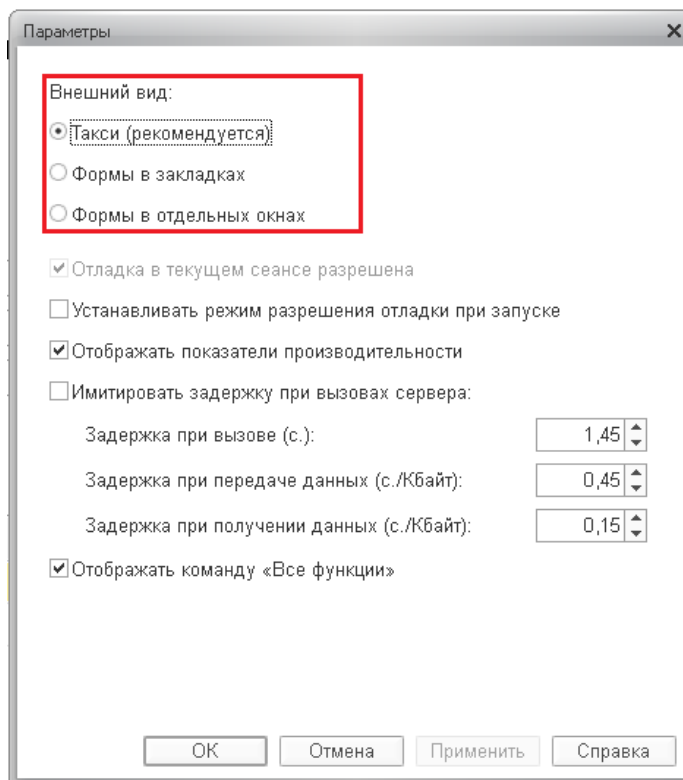
*Версия 8.2* – клиентское приложение работает в интерфейсе 8.2. Переключение в интерфейс Такси невозможно.

*Версия 8.2. Разрешить Такси* – по умолчанию используется интерфейс версии 8.2. Возможно переключение на интерфейс Такси с помощью диалога параметров или командной строки.

*Такси. Разрешить Версия 8.2* – по умолчанию используется интерфейс Такси. Возможно переключение на интерфейс версии 8.2 с помощью диалога параметров или командной строки.

*Такси* – клиентское приложение работает в интерфейсе Такси. Переключение в режим интерфейса 8.2 невозможно.

В режиме 1С:Предприятие.



#### 11.4 Разработка пользовательского интерфейса

При организации пользовательского интерфейса разработчик решает две задачи:

- Реализация форм прикладных объектов и отчетов для ввода и анализа данных
- Реализация интерфейсных элементов приложения: элементов меню, командных панелей, АРМ и т.д.

Для этого в конфигураторе предусмотрены специальные инструменты, которые рассмотрим ниже.

Формы в 1С:Предприятии предназначены для отображения и редактирования информации, содержащейся в базе данных. Формы могут принадлежать конкретным объектам конфигурации или существовать отдельно от них и использоваться всем прикладным решением в целом – общие формы (не принадлежащие конкретным объектам конфигурации).

Основное назначение формы – предоставить пользователю удобное средство для ввода и просмотра информации. Как и бумажный документ, форма позволяет быстро ввести необходимую информацию и запомнить ее для последующей обработки, а при необходимости – вновь вернуться к ранее введенным данным для просмотра или корректировки.

Форма состоит из диалога, модуля и реквизитов формы (полей формы).

Диалог формы представляет собой прямоугольную область экрана, которая, в самом общем случае, содержит различные элементы управления; например, поясняющие надписи, поля ввода информации, элементы управления (например, кнопки) и т.д. С его помощью осуществляется взаимодействие пользователя с программой.

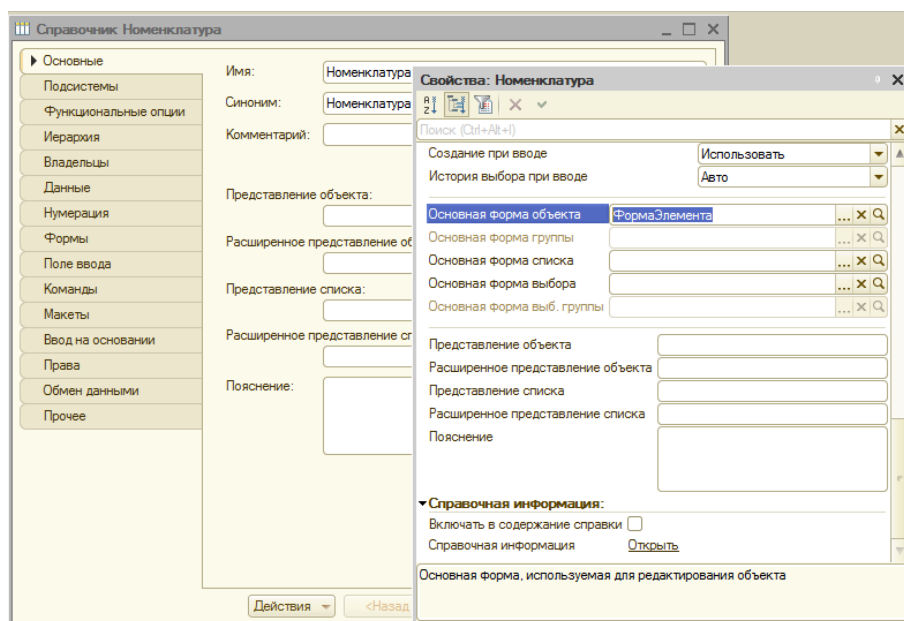
Модуль формы – программа на встроенном языке, отвечающая за работу с элементами управления формы, отработку predefined процедур и выполняющий различные вспомогательные вычисления. С их помощью производится подготовка и обработка реквизитов формы, передача управления к исполнению, а также обработка действий пользователя.

Реквизиты формы – совокупность объектов различных типов, принадлежащих форме. В списке реквизитов есть основной реквизит (выделен жирным шрифтом), который определяет поведение формы при открытии, редактировании или закрытии формы.

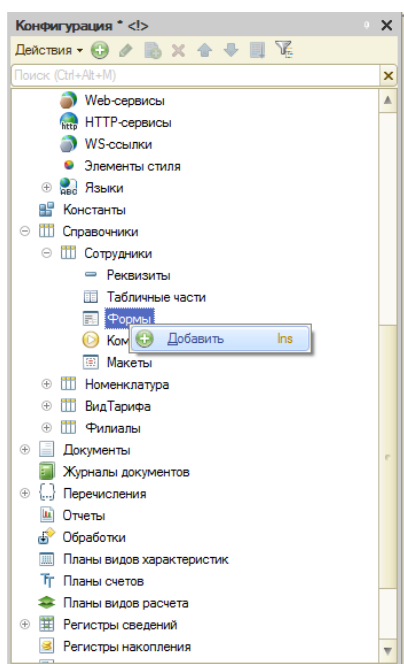
Общие формы – формы, которые доступны из места конфигурации. Обычно общие формы используются для форм настроек программы и других сервисных функций. Например, прикрепление файла к документу, установка ЭЦП, структура подчиненности, установка прав доступа, печать документов, рабочие столы и т.д.

Формы объектов конфигурации. Существует несколько способов создать форму, подчиненную какому-либо объекту конфигурации:

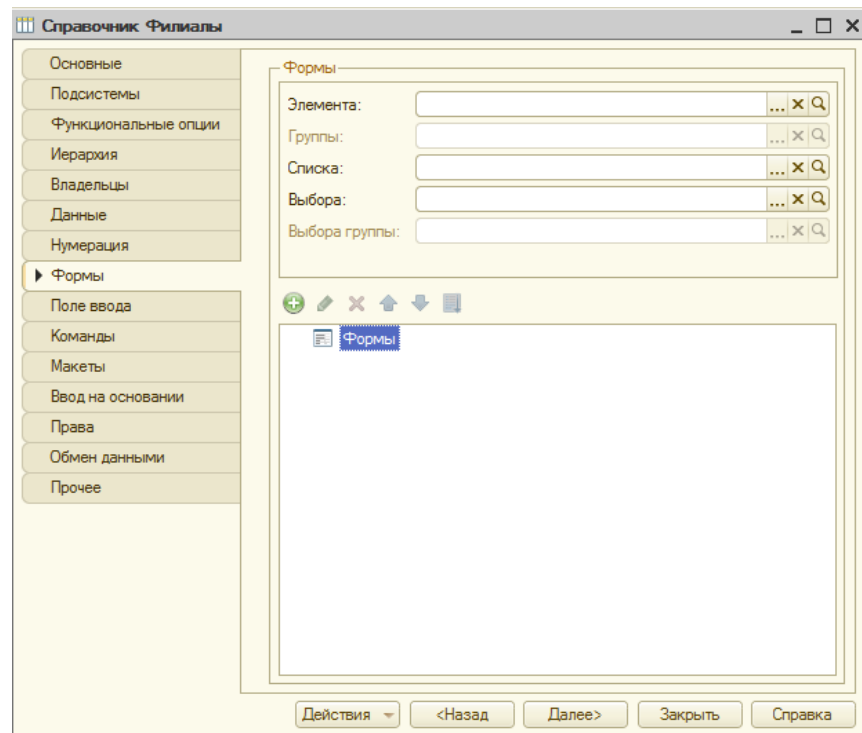
- воспользоваться кнопкой *Открыть* в палитре свойств объекта конфигурации, у нужного типа формы:



- выделить в дереве конфигурации ветвь *Формы объекта*, для которого будет создаваться форма, и воспользоваться либо контекстным меню, либо кнопкой *Добавить* на панели дерева конфигурации:



- открыть окно редактирования объекта конфигурации, перейти на закладку *Формы* и воспользоваться либо кнопкой *Открыть* у необходимого типа формы, либо кнопкой командной панели этого окна *Добавить*, либо контекстным меню в списке форм



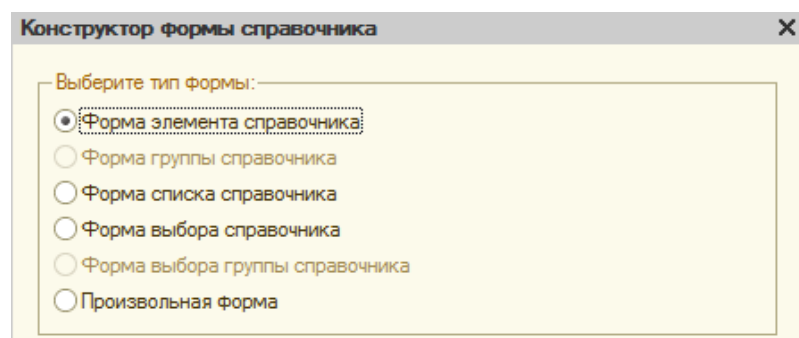
При использовании любого из вышеперечисленных способов будет открыто окно конструктора формы объекта конфигурации.

В зависимости от вида объекта, его свойств конструктором будут предложены характерные для текущего режима создания типы форм (ниже рассмотрены формы изучаемых ранее объектов):

для объекта **СПРАВОЧНИК**:

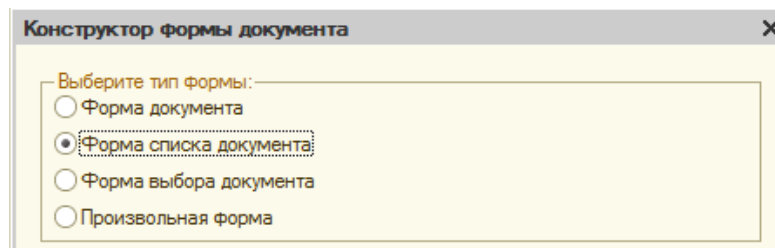
- форма элемента
- форма списка
- форма выбора
- форма элемента

Если справочник иерархический, то возможно создание формы группы и выбора группы



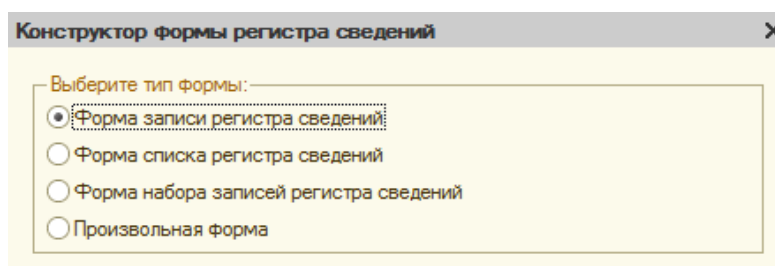
для объекта **ДОКУМЕНТ**

- форма списка
- форма выбора
- форма документа



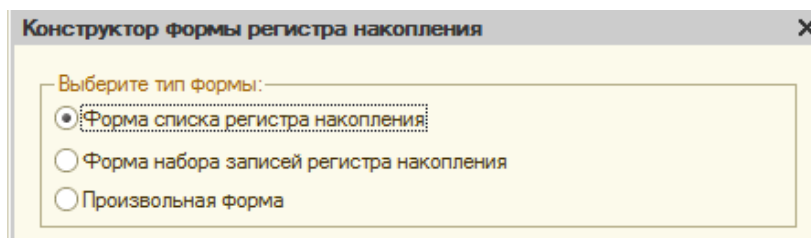
для объекта **РЕГИСТР СВЕДЕНИЙ**

- форма списка
- форма записи
- форма набора записей



для объекта **РЕГИСТР НАКОПЛЕНИЯ**

- форма списка
- форма набора записей



Кроме всех перечисленных форм, для любого из объектов конфигурации можно создать тип формы **Произвольная**. Такая форма после создания не связана ни с какими данными конфигурации. Действия, которые можно будет выполнять в такой форме, целиком и полностью будут зависеть от желания и способностей разработчика формы.

Если у объекта несколько форм одного типа, то одну из них можно выбрать в качестве *основной*. Если при вызове формы объекта не указано явно, какую именно форму следует вызвать, на экран будет выводиться основная форма. Для указания основной формы при создании установите флажок «**Назначить форму основной**».

☒ Назначить форму основной

☐ Основная форма элемента и группы

Имя:

Синоним:

Комментарий:

Реквизиты формы предназначены для хранения данных, с которыми работает форма. В реквизитах хранятся те данные, которые отображаются и редактируются в элементах формы, а также те данные, которые не отображаются, но используются формой в процессе ее работы. Вся совокупность данных, хранящихся в реквизитах формы, называется *данные формы*.

Каждый *элемент формы*, позволяющий изменять данные, связан с некоторым реквизитом формы. Когда пользователь изменяет данные в элементе формы, аналогичные изменения происходят и в связанном с ним реквизите.

Верно и обратное. Когда реквизит формы изменяется программно, изменяется и значение, отображаемое в связанном с ним элементе формы.

Не все реквизиты формы обязательно должны отображаться в форме. Часть реквизитов может быть не связана с элементами формы, а данные, хранящиеся в таких реквизитах, предназначены не для отображения или интерактивного редактирования, а для обеспечения внутренних алгоритмов работы формы или для программного взаимодействия с этой формой из встроенного языка.

Реквизиты формы обладают набором свойств, позволяющих влиять на их поведение. Добавление нового реквизита формы выполняется стандартным способом – с помощью кнопки командной панели.

Среди всех реквизитов формы, как правило, существует один **основной реквизит** (он выделен жирным шрифтом). Он определяет источник данных для формы в целом. Такой реквизит автоматически добавляется конструктором формы, если форма создается подчиненной объекту конфигурации и выбирается вид формы этого объекта, а не произвольная форма. По умолчанию у такой формы после ее создания есть только один реквизит, который и является основным

Реквизит	Используй... всегда	Тип
<b>Объект</b>		(СправочникОбъект.Сот...
Ссылка	<input checked="" type="checkbox"/>	СправочникСсылка.Сотрудн...
Код	<input checked="" type="checkbox"/>	Строка
Наименование	<input checked="" type="checkbox"/>	Строка
ПометкаУдаления	<input checked="" type="checkbox"/>	Булево

Реквизиты    Команды    Параметры

Важнейшим свойством, которое определяет не только то, что за данные будет поставлять форме реквизит, но и как будет выглядеть элемент формы, отображающий эти данные, является Тип

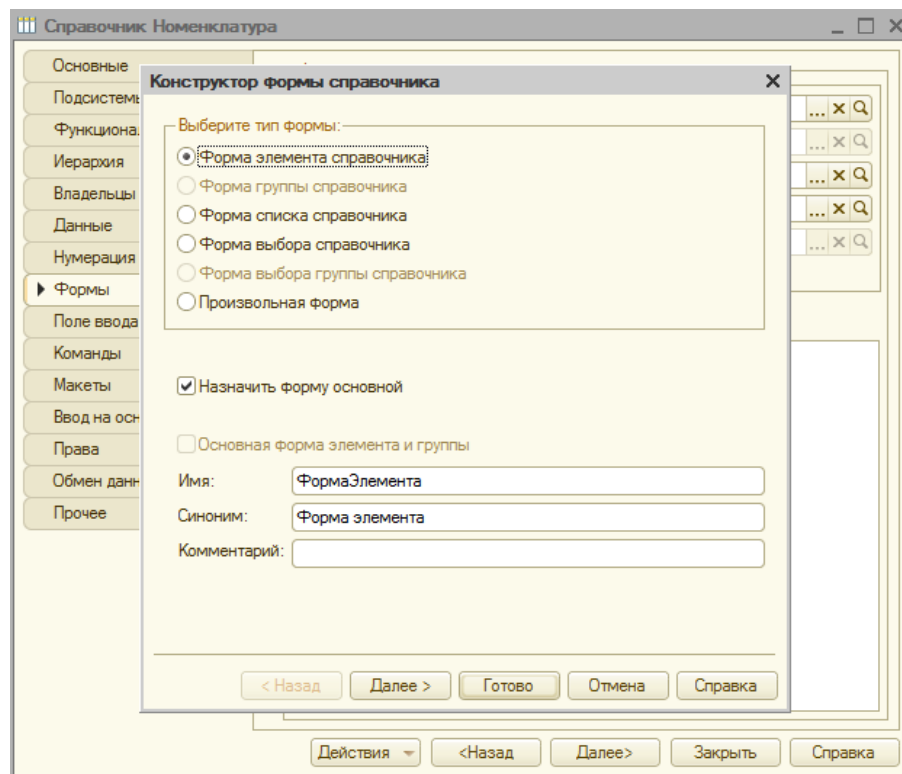
От типа значения основного реквизита формы зависит не только то, какие данные будут отображены в элементах формы, но и поведение самой формы.

Например, если основному реквизиту формы указать тип значения ДокументОбъект.ПриходнаяНакладная, то при закрытии формы



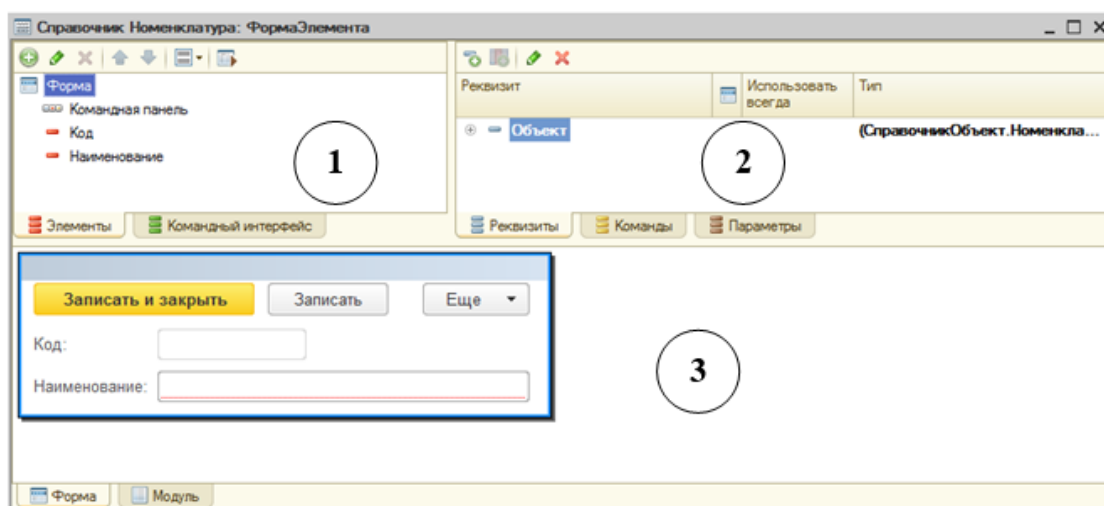
программа будет запрашивать подтверждение записи и проведения документа. Если же основному реквизиту формы указать тип значения СправочникОбъект.Клиенты, то подобного подтверждения при закрытии формы возникать не будет.

Создание новой формы начинается с *конструктора форм*.



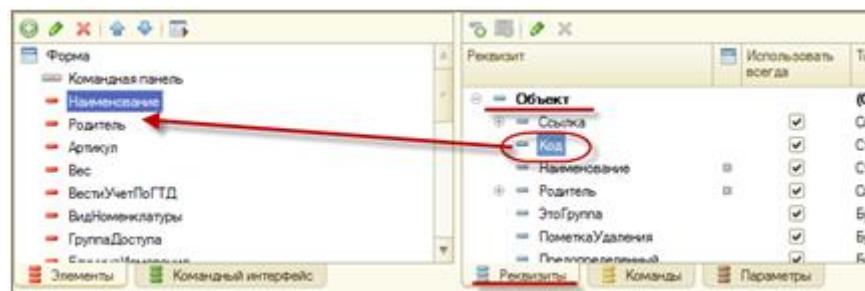
Модификация формы, созданной конструктором, осуществляется в *редакторе формы*.

Окно редактора форм разбито на несколько областей, каждая из которых отвечает за ту или иную функциональность будущей формы.



1. Описывается состав и порядок элементов, из которых состоит форма, команды интерфейса, которые могут выполняться в форме.

2. Описывается состав реквизитов формы, а также команды, выполняемые внутри формы. Реквизит можно перетащить влево и он станет элементом формы (полем на форме).



3. Представлен внешний вид формы, как она может выглядеть на экране пользователя, с учетом описанных реквизитов, элементов, команд формы. При изменении каких-либо настроек в окнах редактора они тут же применяются и изменяют вид формы.

Программный модуль описывает работу формы на встроенном языке.

Для редактирования модуля формы можно воспользоваться закладкой Модуль, которая расположена внизу редактора форм.



В модуле формы располагаются обработчики событий формы, элементов формы, команд формы. Помимо predefined обработчиков событий разработчик прикладного решения может создавать в модуле формы свои процедуры и функции

В форме используются пять элементов:

ГРУППА:

- Обычная группа;
- Группа – Страницы;
- Группа – Командная панель;

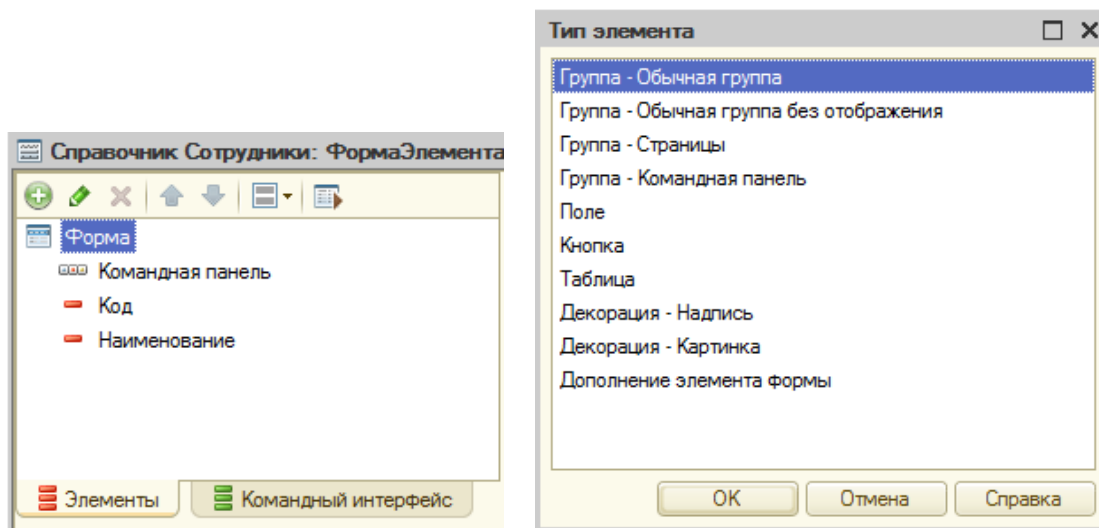
ПОЛЕ;

КНОПКА;

ТАБЛИЦА;

ДЕКОРАЦИЯ:

- Декорация – Надпись;
- Декорация – Картинка.



Элемент формы *Поле* предназначен для отображения примитивных типов данных, текстовых, табличных, HTML-документов, диаграмм, календарей, индикаторов. Тип данных, которые отображает элемент Поле, влияет на то, какие значения может принимать его свойство Вид.

Свойства: Поле

Поиск (Ctrl+Alt+I)

▼ Основные:

Имя	Код
Заголовок	
Вид	Поле ввода
Путь К Данным	Поле ввода
Положение Заголовка	Поле надписи
Видимость	Поле переключателя
Пользовательская видимость	Поле картинки
Доступность	Поле HTML документа
Только Просмотр	Поле текстового документа
Пропускать При Вводе	Авто
Активизировать По Умолчанию	<input type="checkbox"/>
Маска	
Режим Пароля	Авто
Многострочный Режим	Авто
Расширенное Редактирование	Авто

▼ Использование:

Отображение Предупреждения При Редактировании	Авто
---	------

Вид  
Вид. Туре

Элемент формы *Группа* предназначен для группировки других элементов формы, их выделения, для создания командных панелей, подменю, групп кнопок, страниц. Свойство Вид элемента формы Группа может принимать значения:

*Обычная группа.* Элемент формы, предназначенный для логической группировки других элементов формы. Элемент Обычная группа может не отображаться на форме, выполняя при этом функции группировки других элементов. Может отображаться в виде Рамка группы, Линия, Отступ.

*Командная панель.* Элемент формы, предназначенный для группировки кнопок и команд

*Страницы.* Элемент формы, представляющий собой набор страниц.

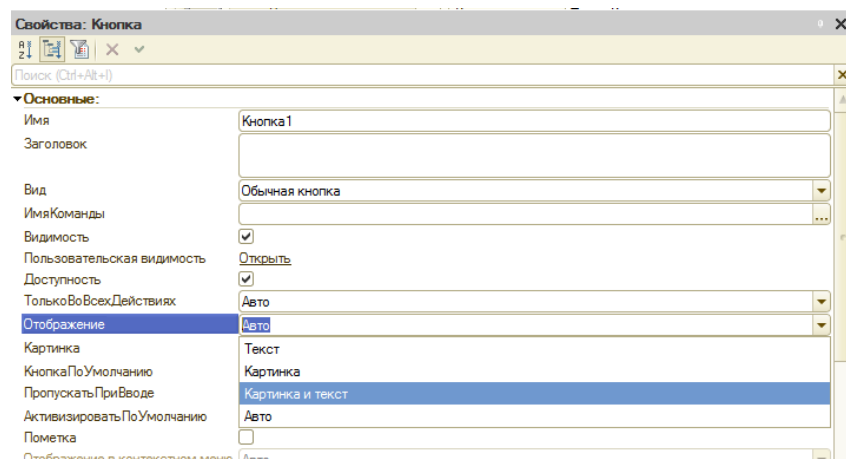
Элемент формы *Кнопка* предназначен для создания кнопок, гиперссылок. Свойство Вид элемента формы Кнопка может принимать значения:

Свойства: Кнопка

Поиск (Ctrl+Alt+I)

▼ Основные:

Имя	Кнопка1
Заголовок	
Вид	Обычная кнопка
Имя Команды	Обычная кнопка
Видимость	Гиперссылка
Пользовательская видимость	Открыть
Доступность	<input checked="" type="checkbox"/>
Только Во Всех Действиях	Авто



Элемент формы *Таблица* предназначен для отображения и редактирования различных табличных данных. У Таблицы могут быть свои командные панели, контекстные меню. Поля колонок элемента Таблица могут принимать различные виды.

Для того чтобы добавить в состав формы новый элемент определенного вида, необходимо воспользоваться одним из этих пяти элементов.

Для отображения различных данных прикладного решения необходимо изменять свойство элемента формы Вид.

## Задание на лабораторную работу

Реализация простых типовых алгоритмов в форме обработки.

№ варианта	Задание_1
1	Дано натуральное число $n$ . Найти все числа меньшие $M_p$ числа Мерсенна. Число Мерсенна - это простое число, представленное в виде $M_p = 2^p - 1$ , где $p$ - тоже простое число.
2	Два натуральных числа называют дружественными, если каждое из них равно сумме всех делителей другого, кроме самого этого числа. Найти все пары дружественных чисел, лежащих в диапазоне от 200 до 300.
3	Дано натуральное число $n$ . Среди чисел 1, 2, ..., $n$ найти все такие, запись которых совпадает с последними цифрами записи их квадрата.
4	Ввести массив, состоящий из 15-ти элементов (двухзначные целые числа). Изменить разрядность цифр, образующих элементы исходного массива и, таким образом, сформировать новый массив. Например, исходный массив: 25 71 84..., новый массив: 52 17 48....
5	Назовем натуральное число палиндромом, если его запись читается одинаково как с начала, так и с конца (пример: 4884, 393, 1, 22). Найти все меньшие 100 натуральных числа, которые при возведении в квадрат дают палиндром.
6	Ввести два массива действительных чисел, состоящих из 7 и 9 элементов. Сформировать третий массив из упорядоченных по убыванию значений обоих массивов.
7	Написать программу поиска суммы последовательности отрицательных чисел, вводимых с клавиатуры, предшествующих первому введенному нулю. Контрольный пример: 1,2,3,-4,5,-2,0.
8	Написать программу поиска произведения последовательности чисел, вводимых с клавиатуры, предшествующих первому введенному отрицательному числу. Контрольный пример: 1,2,3,4,5,-2.
9	Натуральное число называется совершенным, если оно равно сумме всех своих делителей за исключением самого себя. Например. $6 = 1 + 2 + 3$ . Дано натуральное число $n$ . Получить все совершенные числа, меньшие $n$ .
10	Дан массив, состоящий из 12 двоичных чисел. Удалить элементы, которые встречаются более двух раз.
11	Написать программу поиска чисел, лежащих в интервале от -5 до 5, в последовательности чисел, вводимых с клавиатуры, предшествующих первому введенному нулю. Контрольный пример: 1,10,-4,5,-16,-5,0.
12	Написать программу поиска чисел, лежащих в интервале от 3 до 13, в последовательности чисел, вводимых с клавиатуры, предшествующих первому введенному отрицательному числу. Контрольный пример: 1,3,16,7,13,10,2,-1.
13	В массиве хранятся цены на 10 видов мороженого. С помощью датчика случайных чисел заполнить массив целыми значениями, лежащими в диапазоне от 3 до 20 включительно. Определить порядковый номер самого дорогого мороженого.
14	Написать программу, которая изменит введенную строку цифр, разместив сначала цифры занимающие нечетные места, потом четные.

15	Создать процедуру, заменяющую в заданной строке один символ другим. Вводятся три строки. Зашифровать каждую из них, заменив все буквы "с" на "о".
----	---

№ варианта	Задание_2
1	Имеется информация об истории работы пользователей компании в сети интернет. Адреса заданы полным именем до результирующей страницы (пример <a href="http://ixbt.com/testvideo.html">http://ixbt.com/testvideo.html</a> ). Определить наиболее посещаемые сайты. Вывести первых пять адресов сайтов.
2	Задан текст на русском языке. Определить сколько каждая буква встречается в тексте.
3	Во входном файле хранится информация о системе главных автодорог, связывающих города Крыма. Используя эту информацию, постройте дерево, отображающее систему дорог республики с корнем в городе Симферополе.
4	Задан набор путей к различным файлам вида «D:\WORK\Ноябрь\Отчет.xls». Построить в памяти дерево каталогов и вывести на экран.
5	Имеется информация о продажах товаров за некоторый период в виде таблицы: Товар, Количество. Вывести первые пять хорошо продающихся товаров и пять плохо продающихся. В результате строки расположить в порядке убывания.
6	Задан текст, содержащий цитаты известных людей в виде <Цитата>,-<ФИО>. Определить сколько цитат каждого человека содержится в тексте. Вывести полученный результат на экран отсортированный по убыванию частоты вхождения.
7	Задан текст на английском языке. Выделить все различные слова. Для каждого слова подсчитать частоту его встречаемости. Слова, отличающиеся регистром букв, считать одинаковыми. Вывести полученный результат на экран отсортированный по возрастанию частоты вхождения.
8	Имеется таблица, в которой содержится информация об оценках студентов по предмету 1С: ФИО, Предмет, Оценка. Разработать функцию, которая выводит на экран данные сгруппированные по предмету, либо по студенту. Пример вывода: Математика Иванов 5 Петров 4 Программирование Петров 2 Сидоров 5
9	Задан список адресов электронной почты вида <имя_пользователя>@<адрес_сервера>, разделенных «;». Определить количество пользователей для каждого сервера. Вывести на экран первые три популярных сервера.
10	Имеется список поисковых запросов в виде последовательностей слов. Определить первые 5 наиболее встречающихся слов.

11	Имеется таблица, в которой содержится информация об оценках студентов по предмету 1С: ФИО, Предмет, Оценка. Определить средний бал по предмету и по студенту. Вывести на экран обе таблицы.
12	Четырехзначное число, а также число, записанное теми же цифрами в обратном порядке, оба являются точными квадратами. Найти эти числа.
13	Имеется таблица, в которой содержится информация о фильмах: Название фильма, дата и время сеанса, продолжительность сеанса, жанр. Вывести данные о фильмах, начинающихся до 18:00 и продолжительностью сеанса менее 1 часа 30 минут. Вывести на экран обе таблицы.
14	Заданы две одинаковые по длине строки. Построить новую строку, в которой на четных местах расположены элементы первой строки, а на нечетных - элементы второй строки.
15	Дан текст, состоящий из латинских букв. Вывести все гласные буквы, которые не входят более чем в одно слово.

#### Содержание отчета

1. Цель работы.
2. Описание варианта задания.
3. Пошаговое описание процесса выполнения варианта задания.
4. Выводы

#### Контрольные вопросы

1. Расскажите про назначение системы 1С:Предприятие. Основные области применения.
2. Какие типы данных предусмотрены в 1С?
3. Дайте характеристику примитивных типов данных.
4. Перечислите основные функции примитивных типов данных.
5. В каких единицах измерения выполняются вычисления с типом данных «Дата»?
6. Для чего предназначены Универсальные коллекции значений?
7. Перечислите типы, которые представляют собой универсальные коллекции значений. Дайте краткую характеристику каждому из них.
8. Какие методы применимы для универсальных коллекций?
9. Перечислите виды программных модулей. Дайте краткую характеристику каждому из них.
10. Из каких разделов состоит программный модуль? Дайте краткую характеристику каждому разделу.
11. Какие соглашения по именованию переменных, процедур и функций существуют в языке 1С?
12. Какие виды условных операторов предусмотрены в языке 1С?
13. Как организуются циклы в языке 1С?
14. Какие режимы отладки предусмотрены в 1С:Предприятии?
15. Какие обработки предусмотрены в 1С?
16. Как устранить ошибку, возникающую при работе с модальными окнами в 1С?

