Общее описание встроенного языка

Встроенный язык является важной частью технологической платформы 1С:Предприятия 8.0, 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.

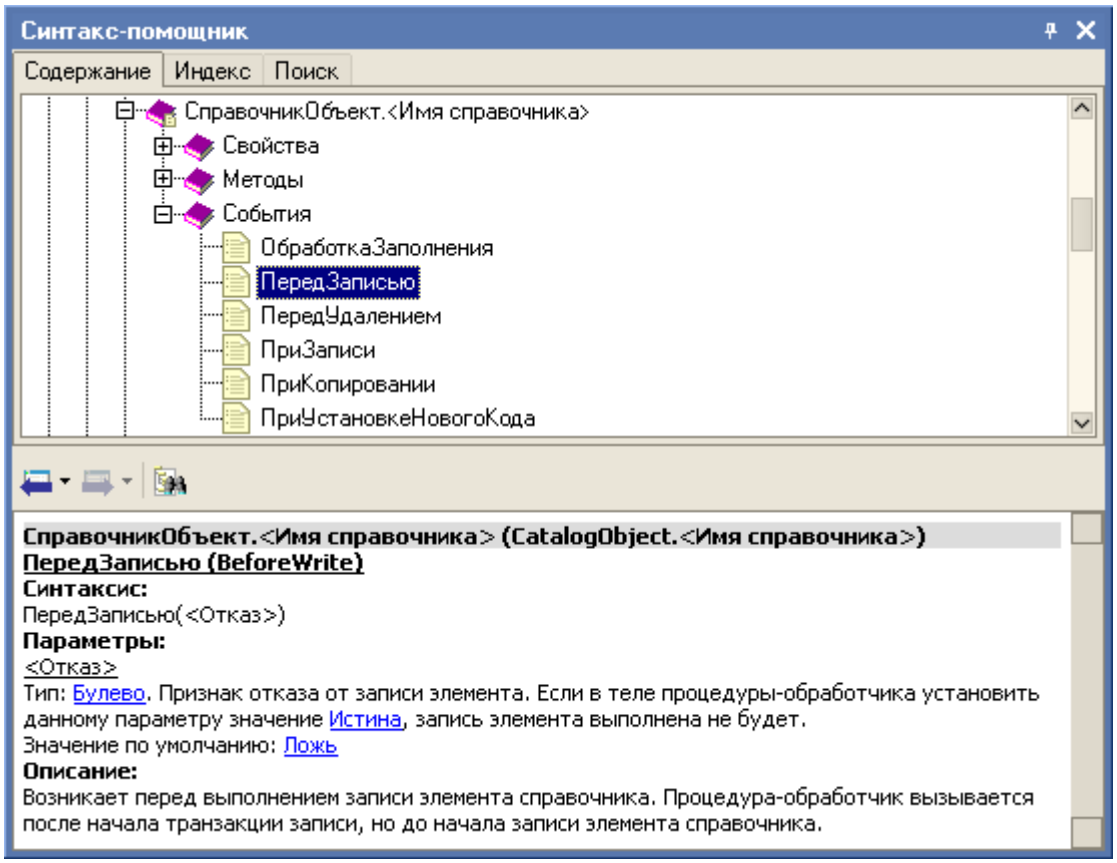
Вот лишь некоторые, наиболее значимые особенности встроенного языка:

- 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;
- отсутствие программного описания объектов конфигурации; разработчик может использовать либо встроенные в платформу объекты, либо объекты, созданные системой в результате визуального конструирования прикладного решения.

Событийная ориентированность встроенного языка. Назначение встроенного языка в системе 1С:Предприятие определяется идеологией создания прикладных решений. Прикладные решения в 1С:Предприятии 8.0 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.

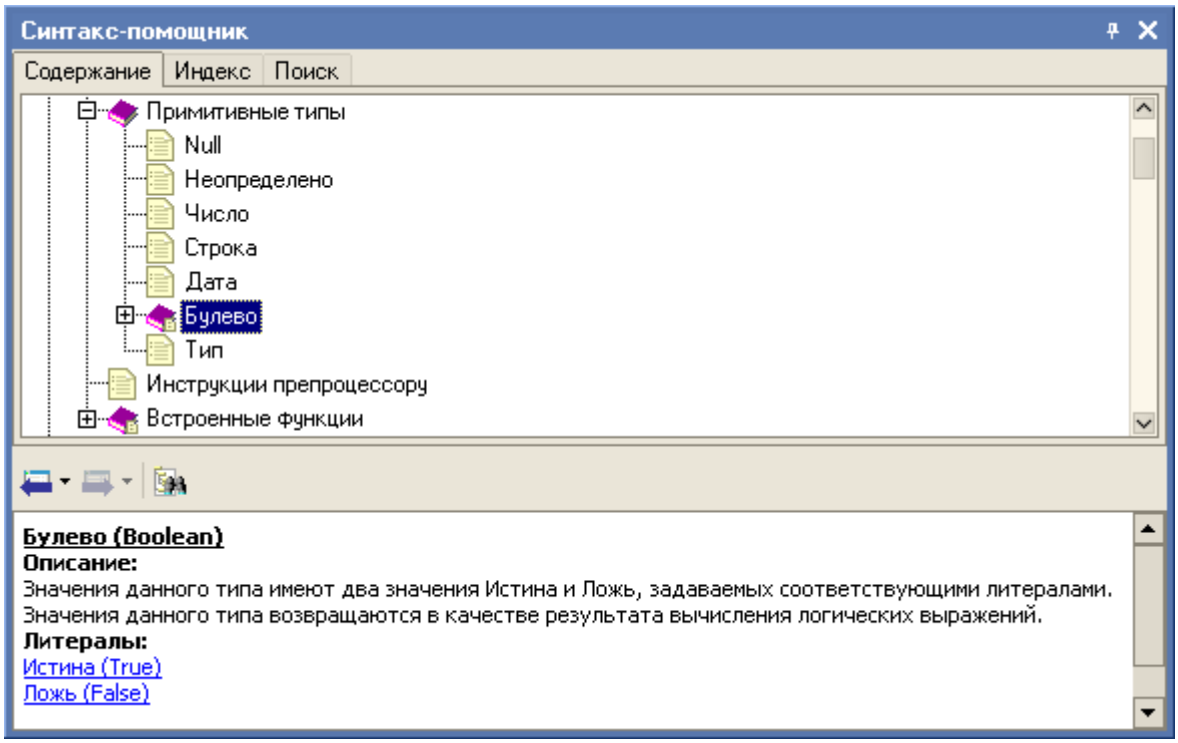
Например, с функционированием объекта прикладного решения Справочник связан ряд событий, среди которых есть событие ПередЗаписью. Это событие возникает непосредственно перед тем, как данные элемента справочника должны быть записаны в базу данных. Разработчик, используя встроенный язык, может описать 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.

Предопределенные типы данных

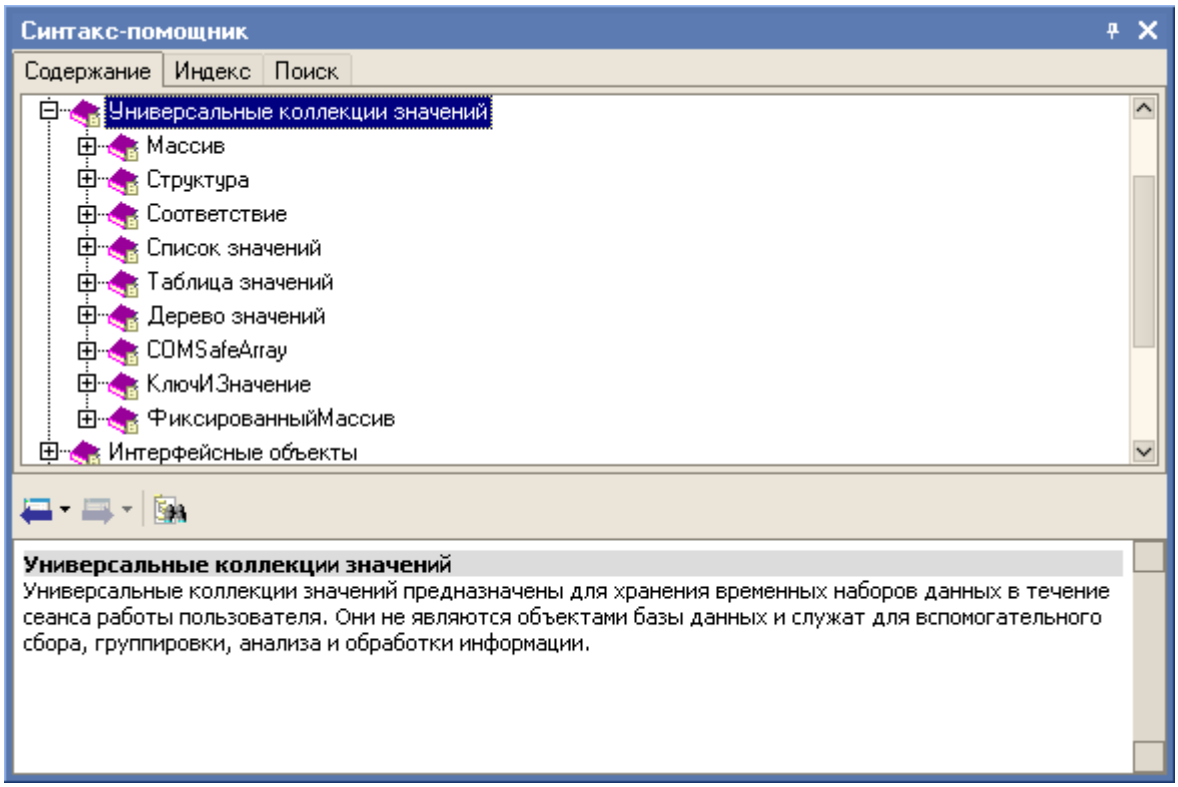
Платформа 1С:Предприятия 8.0 позволяет разработчику использовать различные типы данных.
Существует большое количество типов данных, которые определены на уровне самой платформы. Например, это примитивные типы данных, такие как строка, число, дата и пр.



Описание примитивных типов данных:

- **NULL** - отсутствующее значение. Используется, например, в запросах.
- **Неопределено** - пустое, неопределенное значение. Используется, например, при оценке передачи параметров, в том случае, если при вызове процедуры или функции данный параметр опущен. Реквизиты, имеющие составной тип данных, по умолчанию имеют тип "Неопределено".
- **Булево** - содержит два значения: Истина или Ложь. Используется, например, в логических выражениях - логическое выражение имеет тип "Булево".
- **Дата** - содержит дату и время. По умолчанию имеет значение - 01.01.01 00:00:00 дата начала нашей эры. Время измеряется от начала дня. Запись выражения, имеющего литерал типа "дата", осуществляется следующим образом - '00010101000000'. Сначала записывается год, потом месяц, потом число и потом время. Возможна следующая запись: '20041031'. Время по умолчанию - начало дня.
- **Строка** - бывает переменной, фиксированной и неограниченной длины. В общем случае рекомендуется использовать строки переменной длины.
- **Число** - увеличена разрядность числа до 38 разрядов.
- **Тип** - служит для определения типов значений. Используется, например, для сравнения типов данных. Не имеет литералов и возвращается функциями Тип(<Имя типа>) или ТипЗнч(<Значение>).

Также, существуют более сложные типы данных. Например, платформа поддерживает целый ряд типов, которые представляют собой универсальные коллекции значений: массив, структура, список значений, дерево значений и т.д.



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. Все указанные типы данных создаются только программно.

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.

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.

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.

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.

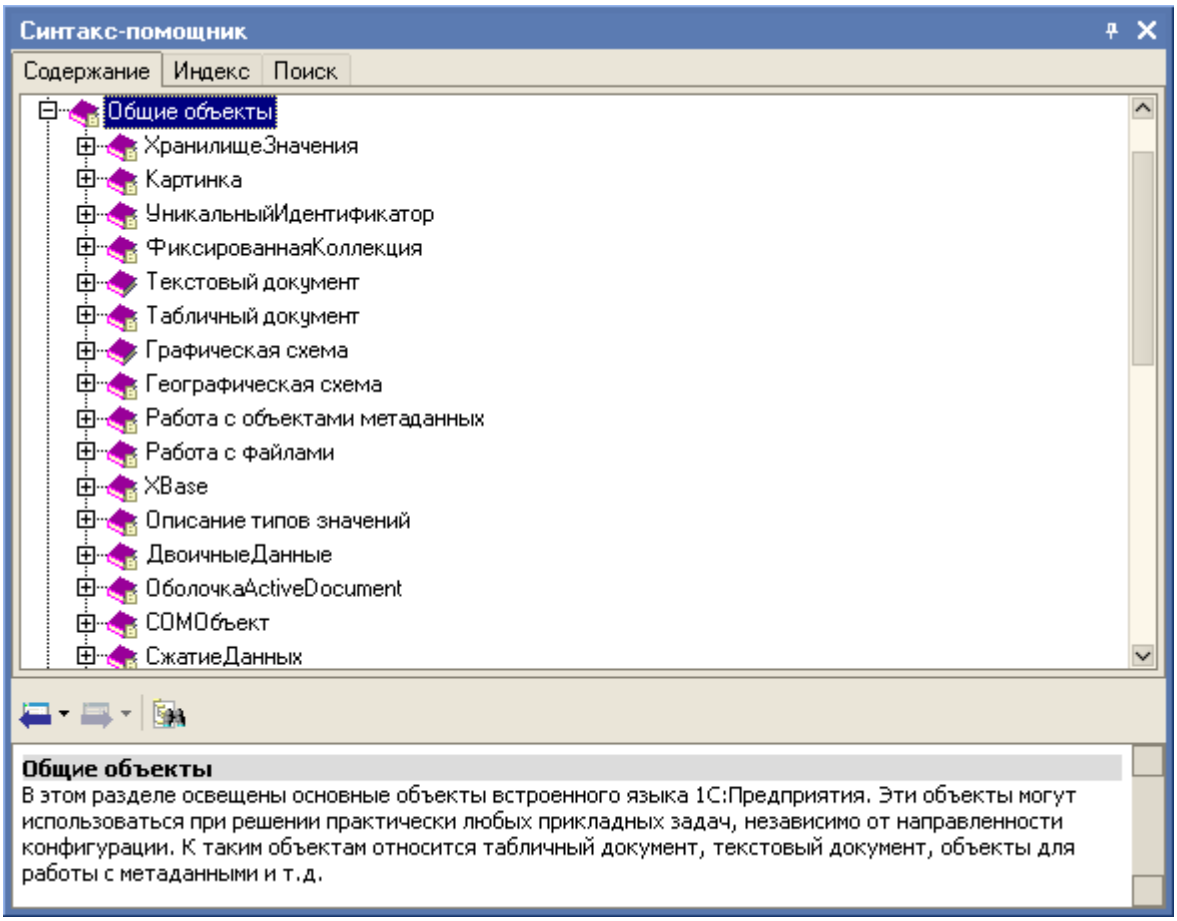
Дерево значений. Дерево значений представляет собой динамически формируемый набор значений любого типа, похожий на таблицу значений. В отличие от таблицы значений,

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.

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

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

Кроме этого в платформе реализованы специфические типы данных, реализующие ту или иную функциональность прикладных решений: текстовый документ, табличный документ, ХранилищеЗначения, ПостроительОтчета, ПостроительЗапроса и пр.



Остановимся подробнее на типе данных "ХранилищеЗначений". Хранилища предназначены для хранения значений, тип которых не может быть выбран в качестве типа поля, например: картинки, двоичные данные. При помещении значения в хранилище значений можно сжимать данные, указав требуемую степень сжатия. Данный метод позволяет уменьшить размер хранимого значения в информационной базе.

К значению, хранящемуся в хранилище, нельзя обращаться напрямую. Перед использованием значения его необходимо извлечь. Не рекомендуется хранить ссылки на объекты, т.к. в системе не поддерживается контроль ссылочной целостности по объектам, сохраненным в полях типа "Хранилище значения".

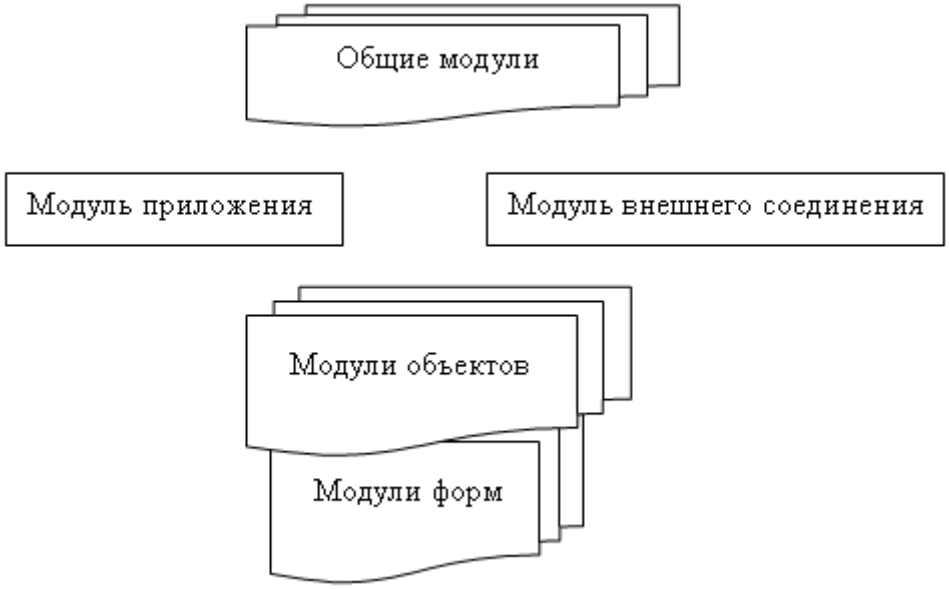
Операции помещения в хранилище и извлечения из него могут занимать продолжительное время, что необходимо учитывать при проектировании структур объектов.

Пример использования хранилища значения на примере справочника "Картинки":

```
Процедура ОтобразитьКартинку()
ЗначениеКартинки = Картинка.Получить();
Если ЗначениеКартинки <> Неопределено Тогда
ЭлементыФормы.ПолеКартинки.Картинка = ЗначениеКартинки;
Иначе
ЭлементыФормы.ПолеКартинки.Картинка = Новый Картинка(Неопределено);
КонецЕсли;
КонецПроцедуры
Процедура ВыбратьИзФайлаНажатие(Элемент
// Выбор файла с просмотром
ДиалогВыбораФайла = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Открытие);
ДиалогВыбораФайла.Каталог = "";
ДиалогВыбораФайла.ПредварительныйПросмотр = Истина;
ДиалогВыбораФайла.ИндексФильтра = 0;
Если ДиалогВыбораФайла.Выбрать() Тогда
Файл = Новый Файл(ДиалогВыбораФайла.ПолноеИмяФайла);
Картинка = Новый ХранилищеЗначения(Новый
Картинка(ДиалогВыбораФайла.ПолноеИмяФайла));
ОтобразитьКартинку();
КонецЕсли;
КонецПроцедуры
```

Программные модули

Схема 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. Например, модули формы привязаны к формам объектов и позволяют детально описывать реакцию формы на действия пользователя.

В конфигурации существует несколько видов модулей. Это модуль приложения, модуль внешнего соединения, общие модули, модули форм и модули объектов конфигурации (менеджеров значения констант, справочников, документов, планов видов характеристик, планов счетов, планов видов расчета, планов обмена, бизнес-процессов, задач, отчетов, обработок, наборов записей регистров).

Если конфигурация запускается не в режиме клиентской сессии, а в режиме COM-соединения, то вместо модуля приложения используется модуль внешнего соединения.

Для написания и редактирования текстов программных модулей предназначен редактор текстов и модулей. Тексты программных модулей могут содержать конструкции как на русском, так и на английском языках в любой комбинации.

Разделы программного модуля в порядке их размещения:

- Раздел объявления переменных;
- Раздел описания процедур и функций;
- Раздел основной программы.

Ограничения программных модулей:

- Общие модули содержат только раздел описания процедур и функций.

В общих модулях описываются такие процедуры и функции, алгоритмы которых неоднократно используются в других модулях.

В модуле приложения описываются такие процедуры и функции, как "ПриНачалеРаботыСистемы", "ПриЗавершенииРаботыСистемы" и др.

В модуле внешнего соединения описываются специфические процедуры и функции, которые актуальны для использования в режиме внешнего соединения. В режиме Com-соединения по сути модуль приложения заменяется на модуль внешнего соединения.

В модуле объекта описываются такие процедуры и функции, как "ПриЗаписи", "ПриУстановкеНовогоКода", "ПриКопировании", "ОбработкаЗаполнения" и др.

В модуле формы описываются такие процедуры и функции, как "ПриОткрытии", "ПриЗакрытии", "ОбработкаВыбора", "ВнешнееСобытие" и др.

Для того, чтобы переменная, процедура или функция были доступны в других модулях (в соответствии с контекстом использования модулей) необходимо в конце строки объявления написать ключевое слово "Экспорт". Использование ключевого слова "Экспорт" имеет смысл только для тех переменных, которые описаны в разделе описания переменных программного модуля. Например:

```
Перем глТекущийПользователь Экспорт
Процедура Пересчет() Экспорт
```

Процедура или функция, объявленные с ключевым словом "Экспорт" в модуле объекта, дополняют контекст этого объекта. Например:

```
Справочники.Клиенты.НайтиПоКоду(ВыбКод,,,).ПолучитьОбъект().Печать();
```

При этом процедура Печать() объявлена в модуле объекта справочника "Клиенты" с ключевым словом "Экспорт".

Отличие процедур и функций состоит в том, что функция имеет возвращаемое значение. Параметры процедур и функций по умолчанию передаются по ссылке. Для того, чтобы передать параметр по значению используется ключевое слово "Знач". Например:

```
Процедура Пересчет(Количество, Сумма, СтавкаНДС, Знач ФлагРасчетаСтавкиНДС)
```

Текст программного модуля может содержать однострочные комментарии, которые начинаются с комбинации символов //.

Например:

```
// данная процедура предназначена для расчета суммы по строке
```

Задание 7

1. Создайте общий модуль. Создайте в общем модуле процедуру Пересчет() с ключевым словом Экспорт.
2. Предусмотрите вызов процедуры Пересчет() из форм документов "Поступление материалов" и "Акт об оказании услуг".

Встроенный язык. Переменные

Для объявления переменных используется ключевое слово "Перем". Существует также неявное объявление переменных при первом присвоении значения переменной.

Например:

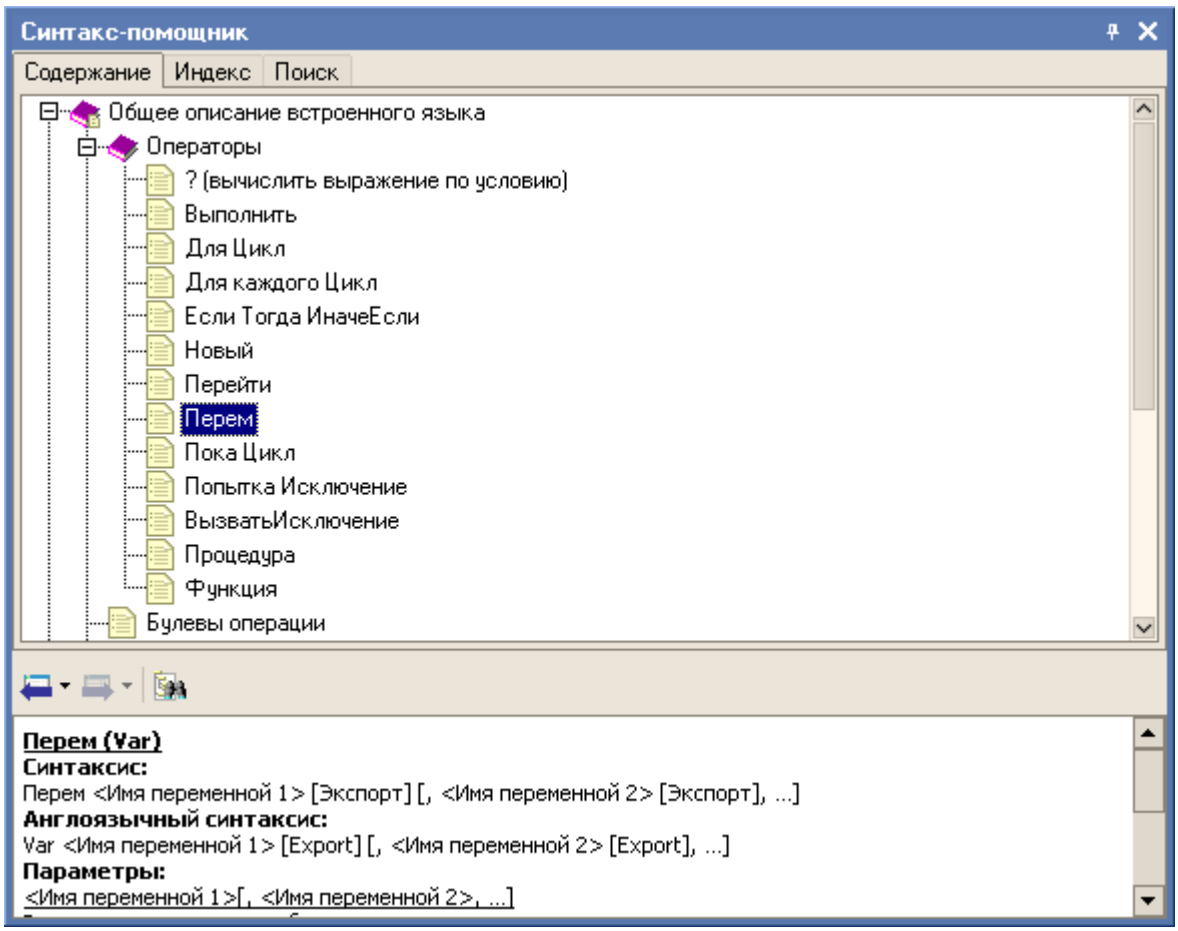
```
Перем Значение1;
Значение2 = 4;
```

Переменная Значение1 объявлена явно, а переменная Значение2 объявлена неявно. При неявном объявлении переменной система определяет ее тип по присваиваемому ей значению. При повторном присвоении переменной значения ее тип может измениться.

В именах переменных допускается использование символов и цифр, а также символа подчеркивания. При этом имя переменной должно начинаться только с символа. Использование пробелов в именах переменных не допускается.

Встроенный язык. Операторы

Операторы имеют вид стандартного обращения к процедуре, за исключением оператора присваивания (А = В;) и синтаксических конструкций встроенного языка (например, таких как Для, Пока, Если). Между собой операторы обязательно следует разделять символом ";" (точка с запятой). Конец строки не является признаком конца оператора, т.е. операторы могут свободно переходить через строки и продолжаться на другой строке. Можно располагать произвольное число операторов в одной строке, разделяя их символом ";".



Операторы языка в программном модуле можно подразделить на две категории: операторы объявления переменных и исполняемые операторы.

Операторы объявления переменных создают имена переменных, которыми манипулируют исполняемые операторы.

Любой исполняемый оператор может иметь метку, используемую в качестве точки перехода в операторе "Перейти".

В общем случае формат оператора языка следующий:

~метка: Оператор[(Параметры)][ДобКлючевоеСлово];

Одним из операторов являются циклы. Во встроенном языке различают следующие виды циклов:

- Пока <условие> цикл.
- Для каждого <имя переменной> из <имя коллекции> цикл.
- Для <имя переменной> = <начальное значение> по <конечное значение> цикл.

Для досрочного прекращения цикла используется оператор "Прервать".

Пример использования цикла "для каждого":

```
// Создание массива
Массив = Новый Массив(10);

// Заполнение массива
Для Сч=0 по 9 Цикл
Массив[Сч]=Сч;
КонецЦикла;

// Индикация массива
Для Каждого Элемент из Массив Цикл
Сообщить(Элемент);
КонецЦикла;
```

Рассмотрим следующий вид операторов - "условия". Условия оформляются следующим образом:

```
Если <ключевое слово > (<логическое выражение>) <ключевое слово> (<логическое выражение>) :
:
ИначеЕсли <ключевое слово> (<логическое выражение>) <ключевое слово> (<логическое выражение>) :
:
Иначе
КонецЕсли;
```

В записи условий могут использоваться следующие ключевые слова: "И", "ИЛИ", "НЕ". Ветвей "ИначеЕсли" может быть неограниченное количество или ни одной.

Логические выражения имеют тип Булево. При этом, если в условии используется несколько логических выражений, соединенных ключевыми словами "И" или "ИЛИ", и по значению первого логического выражения можно определить значение всего условия, то вычисление значений остальных логических выражений не производится. Например:

```
Значение1 = 5;
Значение2 = 4;
Если (Значение1 = 5) или (Значение2 = 4) тогда
```

В данном случае в условии используется ключевое слово "ИЛИ". Поэтому, если значение первого логического выражения Истина, то вычисление второго логического выражения произведено не будет.

Пример укороченной записи логического выражения:

```
Значение1 = Истина;
Значение2 = Ложь;
Если Значение1 И НЕ Значение2 тогда
```

Встроенный язык. Инструкции препроцессора

Код системы 1С:Предприятие 8.0 может исполняться в файловом, клиентском и серверном окружении, а также в сессии COM-соединения. При этом можно в конфигураторе настроить место выполнения (на сервере или на клиенте) различных процедур и функций для каждого из вариантов.

Для указания разрешения использования процедур следует воспользоваться директивой препроцессора.

Конструкция типа

```
#Если Сервер тогда
:
Процедура Проц1() экспорт
:
КонецПроцедуры
:
#КонецЕсли
```

позволит указать системе, что процедура Проц1() должна выполняться на сервере, а конструкция

```
#Если Клиент тогда
:
Процедура Проц2() экспорт
:
КонецПроцедуры
:
#КонецЕсли
```

укажет на выполнение Проц2() на клиентской машине.

Для включения использования процедур и функций в сессии внешнего соединения применяется инструкция препроцессора

```
#Если ВнешнееСоединение тогда
:
#КонецЕсли
```

Встроенный язык. Системные перечисления

Системные перечисления предназначены для определения некоторого набора предопределенных значений. Доступ к системным перечислениям осуществляется как к свойствам глобального контекста. Конкретные значения указываются через точку от имени системного перечисления. Значения системных перечислений не перебираются по индексу.

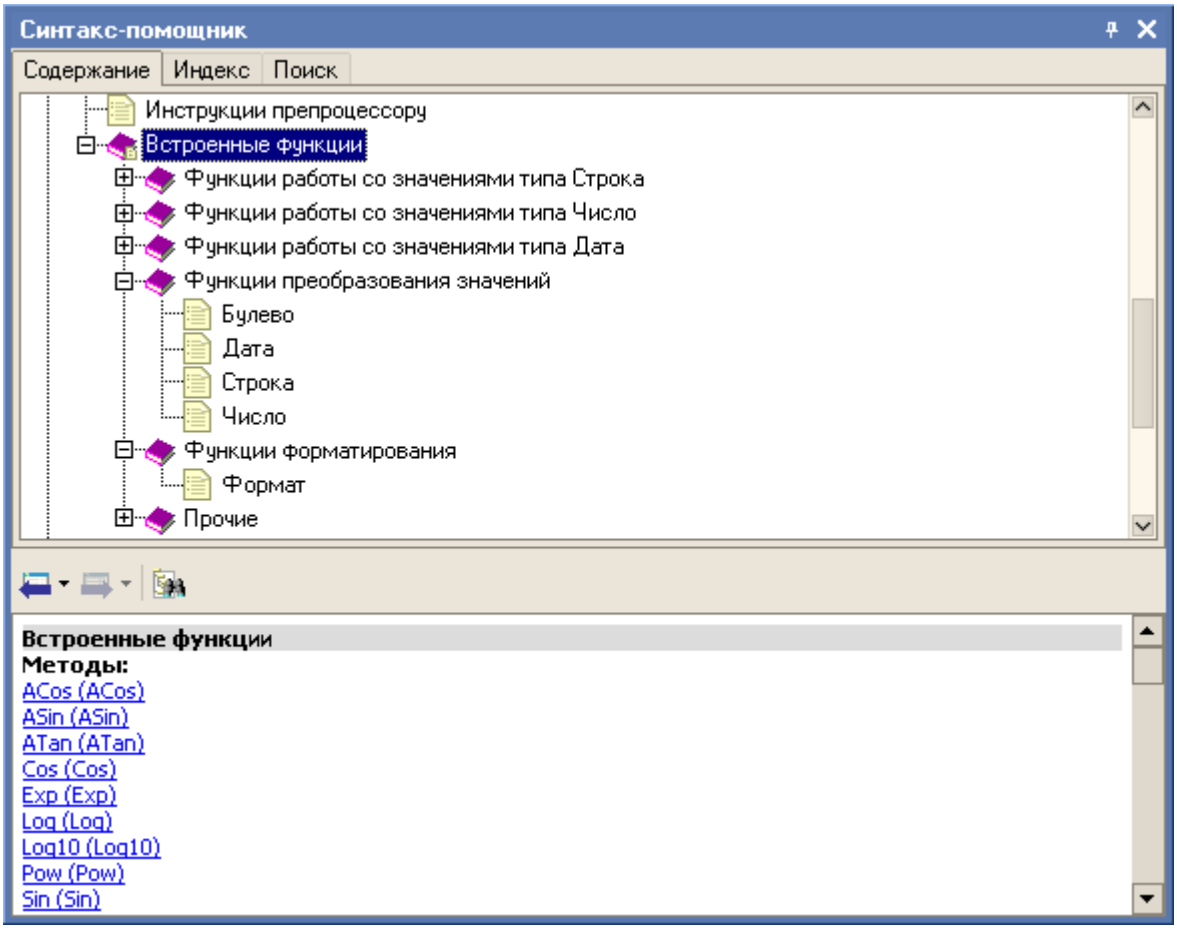
К системным перечислениям, например, относятся следующие:

- КодВозвратаДиалога
- РежимБегущейСтроки
- ОбходРезультатаЗапроса
- РежимПроведенияДокумента
- ВариантПериода
- и др.

Не следует путать системные перечисления и системные наборы значений (например системный набор значений "Символы").

Встроенный язык. Встроенные функции

К встроенным функциям языка относятся функции работы со значениями типа дата, строка, число, а также функции преобразования значений и функции форматирования (Формат()).



Для работы с переменными типа "дата" в платформе предусмотрены следующие встроенные функции языка:

- День(<дата>)
- ДеньГода(<дата>)
- ДеньНедели(<дата>)
- ДобавитьМесяц(<дата>,<число>)
- КонецГода(<дата>)
- КонецДня(<дата>)
- КонецЧаса(<дата>)
- ТекущаяДата() и др.

Задание 8

Создайте внешнюю обработку "ПримерРаботыСДатой". На форме обработки разместите реквизиты: "ИсходнаяДата" и "КоличествоДней". По кнопке сформировать организуйте вывод в текстовое поле даты, полученной прибавлением к исходной дате указанного количества дней.

Встроенный язык. Глобальный контекст

Глобальный контекст инициализируется при открытии конфигурации в режиме "1С:Предприятие" и существует вплоть до ее закрытия. Все свойства, процедуры и функции глобального контекста доступны в любом программном модуле конфигурации. Доступ к свойствам, процедурам и функциям глобального контекста осуществляется непосредственно из любого программного модуля, используя их имена (без ссылки на какой-либо объект).

Важно! У всех событий глобального контекста назначены предопределенные процедуры-обработчики. Имена процедур соответствуют именам событий. Все процедуры должны располагаться в модуле приложения.

Глобальный контекст:
Свойства:

БиблиотекаКартинок (PictureLib)
БиблиотекаСтилей (StyleLib)
БизнесПроцессы (BusinessProcesses)
ВнешниеОбработки (ExternalProcessings)
ГлавныйИнтерфейс (MainInterface)
ГлавныйСтиль (MainStyle)
Документы (Documents)
ЖурналыДокументов (DocumentJournals)
Задачи (Tasks)
ИспользованиеРабочейДаты (WorkingDateUse)
:
ПользователиИнформационнойБазы (InfoBaseUsers)
Последовательности (Sequences)
РабочаяДата (WorkingDate)
РегистрыБухгалтерии (AccountingRegisters)
РегистрыНакопления (AccumulationRegisters)
РегистрыРасчета (CalculationRegisters)
РегистрыСведений (InformationRegisters)
Справочники (Catalogs)

В разделе "Интерфейсы" мы уже упоминали о программном переключении интерфейсов. Рассмотрим подробнее свойство глобального контекста "ГлавныйИнтерфейс". Тип значения: КоллекцияЭлементовУправленияИнтерфейсами. Используется для доступа к определенным в конфигурации пользовательским интерфейсам. Недоступен на сервере 1С:Предприятие. Не используется в модуле внешнего соединения.

Свойство "РабочаяДата" содержит рабочую дату, используемую в текущем сеансе работы с конфигурацией. Доступно для записи в случае, если свойство "ИспользованиеРабочейДаты" имеет значение "Назначать".

У КоллекцияЭлементовУправленияИнтерфейсами есть метод

ПереключитьИнтерфейс(<Имена интерфейсов>)

<Имена интерфейсов> (необязательный) - тип: Строка, перечисленные через запятую имена интерфейсов, которые следует сделать видимыми.

Это способ группового управления видимостью командных интерфейсов. При вызове данного метода сначала становятся невидимыми все интерфейсы, кроме имеющих значение Ложь у свойства Переключаемый, а затем становятся видимыми перечисленные в параметре интерфейсы.

Например:

```
Интерфейсы.ПереключитьИнтерфейс("ИнтерфейсОсновной,ИнтерфейсПродажи");
```

Методы:

Base64Значение (Base64Value)

Base64Строка (Base64String)

XMLЗначение (XMLValue)

XMLСтрока (XMLString)

XMLТип (XMLType)

XMLТипЗнч (XMLTypeOf)

ВвестиДату (InputDate)

ВвестиЗначение (InputValue)

ВвестиСтроку (InputString)

ВвестиЧисло (InputNumber)

ВозможностьЧтенияXML (CanReadXML)

Вопрос (DoQueryBox)

:

УдалитьОбъекты (DeleteObjects)

УдалитьФайлы (DeleteFiles)

УстановитьЗаголовокСистемы (SetCaption)

УстановитьМонопольныйРежим (SetExclusiveMode)

ЧислоПрописью (NumberInWords)

Рассмотрим подробнее процедуру "ОбработкаПрерыванияПользователя". Данная процедура предназначена для прерывания работы встроенного языка при нажатии пользователем клавиши Ctrl+Break. Метод проверяет, была ли нажата пользователем клавиша Ctrl+Break. Если клавиша была нажата, то выполнение встроенного языка прекращается и выдается соответствующее сообщение. Данный метод рекомендуется использовать в длительных циклических операциях. Метод будет иметь действия только в тех случаях, когда допускается прерывание выполнения модулей. Прерывание выполнения допускается, если оно инициировано определенным интерактивным действием пользователя. К таким действиям относятся: нажатие кнопки в форме; выбор пункта меню или кнопки панели инструментов в форме; выбор пункта меню или кнопки панели инструментов интерфейса; действия, инициируемые обработчиками событий "Выбор" элементов управления.

Свойства и методы глобального контекста не являются конструкциями встроенного языка.

События:
ОбработкаВнешнегоСобытия (ExternEventProcessing)
ПередЗавершениемРаботыСистемы (BeforeExit)
ПередНачаломРаботыСистемы (BeforeStart)
ПриЗавершенииРаботыСистемы (OnExit)
ПриЗавершенииРаботыСистемы (OnExit)
ПриНачалеРаботыСистемы (OnStart)
ПриНачалеРаботыСистемы (OnStart)

Задание 9

Перед завершением работы системы задать вопрос: "Вы уверены, что хотите завершить работу с системой?".

Типообразующие объекты, их свойства, методы и события

Наряду с типами данных, которые определены на уровне платформы, конкретное прикладное решение может использовать уникальные типы данных, существующие только в этом конкретном прикладном решении. Причем технологическая платформа 1С:Предприятия 8.0 будет полностью поддерживать работу с этими типами данных точно так же, как и с типами, которые определены на уровне самой платформы.

Как правило, появление новых типов данных в прикладном решении связано с использованием прикладных объектов. На уровне технологической платформы поддерживается несколько классов прикладных объектов, которые сами по себе не могут быть использованы в конкретном прикладном решении. Например, можно перечислить

такие классы прикладных объектов как Справочники, Документы, Регистры сведений, Планы видов характеристик и пр.

Для каждого класса прикладных объектов определена соответствующая ему базовая функциональность: типы таблиц базы данных, которые должны быть созданы для хранения данных, типовые формы, типовые объекты языка, наборы прав и пр.

Разработчик, создавая прикладное решение, не имеет возможности использовать эти классы напрямую, однако может добавить в свое прикладное решение новый объект конфигурации, наследующий всю функциональность того или иного класса.

Например, разработчик может добавить в свое прикладное решение новый справочник Номенклатура, который будет наследовать функциональность класса Справочники, или новый документ КассовыйОтчет, который будет наследовать функциональность класса Документы.

Сразу же после такого добавления разработчику становятся доступны новые типы данных, состав которых определяется принадлежностью объекта к тому или иному классу прикладных объектов.

Например, после создания нового справочника Номенклатура, становятся доступны следующие типы данных:

- СправочникМенеджер.Номенклатура;
- СправочникСсылка.Номенклатура;
- СправочникОбъект.Номенклатура;
- СправочникВыборка.Номенклатура;
- СправочникСписок.Номенклатура.

Еще один момент, на котором следует акцентировать внимание, проще всего продемонстрировать на примере.

Допустим, в прикладном решении созданы два новых справочника: Номенклатура и Цены. Несмотря на то, что оба эти объекта унаследовали функциональность соответствующего класса Справочники, и для них в прикладном решении был создан один и тот же состав типов данных, одноименные типы данных будут являться различными типами данных. Например, СправочникОбъект.Номенклатура и СправочникОбъект.Цены - это различные типы данных.

Так происходит потому, что разработчик может дополнительно к базовой функциональности, унаследованной от соответствующего класса, добавить свою, особенную для каждого объекта конфигурации. Например, оба упомянутых выше справочника могут содержать табличные части (это унаследовано от класса Справочники). Однако для справочника Цены разработчик не создаст ни одной табличной части, в то время как для справочника Номенклатура он создаст, например, три табличные части. Очевидно, что структура хранения данных типа СправочникОбъект.Номенклатура будет значительно отличаться от структуры хранения данных типа СправочникОбъект.Цены.

Для работы с такими типами данных (классами прикладных объектов) в системе реализована единая схема работы через объекты ":Менеджер". Объекты типа КонстантыМенеджер, СправочникиМенеджер, ДокументыМенеджер,

ОбработкиМенеджер и аналогичные им предназначены для обращения к коллекции значений соответствующих менеджеров объектов конфигурации. Свойствами этих объектов являются другие объекты типа КонстантаМенеджер, СправочникМенеджер, ДокументМенеджер, которые, в свою очередь, предоставляют доступ к конкретным объектам (справочник, документ, константа).

Например:

СправочникиМенеджер - объект, обеспечивает доступ к менеджерам всех справочников конфигурации.

ВсеСправочники=Справочники; //в данном случае тип переменной ВсеСправочники будет СправочникиМенеджер.

"Справочники" - это свойство глобального контекста, с помощью которого идет обращение к менеджеру справочников конфигурации.

СправочникМенеджер.<имя> - объект, обеспечивает доступ к конкретному справочнику конфигурации.

СправочникКлиенты=Справочники.Клиенты; //тип переменной СправочникКлиенты будет СправочникМенеджер.

СправочникОбъект.<имя> - объект, обеспечивает доступ к конкретному элементу справочника, его свойствам и методам.

ЭлементСКодом1 = Справочники.Клиенты.НайтиПоКоду(1).ПолучитьОбъект(); //тип переменной ЭлементСКодом1 будет СправочникОбъект.

СправочникСписок.<Имя> - объект, предназначен для управления списком элементов справочника, отображаемых в табличном поле. Данный объект используется для визуального представления списка справочника.

Такая схема обращения является единой для всех объектов конфигурации.

Для выбора множества объектов используются объекты типа "СправочникВыборка", "ДокументВыборка" и т.д.

ВыборкаСправочника = Справочники.Клиенты.Выбрать(); //тип переменной ВыборкаСправочника будет СправочникВыборка.

Для объекта "КонстантыМенеджер" в системе предусмотрен метод СоздатьНабор() с возвращаемым значением типа "КонстантыНабор". Объект "КонстантыНабор" предназначен для управления набором указанных в параметрах метода СоздатьНабор() констант. Используя данный объект, можно прочитать и записать в одной транзакции значения для выбранных констант.

```
Набор = Константы.СоздатьНабор("Руководитель, ИНН");
Набор.Прочитать();
Набор.Руководитель = "Иванов И.И.";
Набор.ИНН = "1234567890";
Набор.Записать();
```

Для объектов "РегистрСведенийМенеджер", "РегистрНакопленияМенеджер", "РегистрБухгалтерииМенеджер", "РегистрРасчетаМенеджер" предусмотрен метод СоздатьНаборЗаписей() с возвращаемыми значениями РегистрСведенийНаборЗаписей, РегистрНакопленияНаборЗаписей и т.д.

К свойствам, реквизитам и методам описанных выше объектов обращаемся через точку. Например:

```
Код = Справочники.Контрагенты.НайтиПоКоду(1).Код;
//в данном примере мы обратились к методу справочника "НайтиПоКоду()"
//и к реквизиту элемента справочника "Код".
```

Рассмотрим подробнее работу с набором записей регистра сведений. Основная задача регистра сведений - хранить информацию, которая развернута по определенной комбинации измерений и, при необходимости, по времени. Из принципов работы регистра сведений вытекает то, что в системе может быть только одна запись с определенным набором и периодом измерений. Уникальность записей по набору измерений принципиально отличает регистры сведений от регистров накоплений, которые позволяют вводить несколько записей с одинаковым значением измерений и периодом. Строки регистра сведений, содержащие информацию о значениях ресурсов для определенных значений измерений, конкретного периода, регистратора, называются записями. Для считывания и занесения набора записей в базу данных служит объект "РегистрСведенийНаборЗаписей".

Некоторые методы объекта "РегистрСведенийНаборЗаписей":

- Добавить() - добавляет новую запись в набор.
- Количество() - получает количество записей в наборе.
- Прочитать() - считывает записи из базы данных по установленному отбору.
- Удалить() - удаляет запись из набора записей регистра сведений.
- Записать(<замещать>) - записывает набор записей в базу данных. В зависимости от переданного параметра, может быть выполнено добавление записей или их замещение. Если для регистра сведений, подчиненного регистратору, выполняется добавление записей, то после выполнения записи набор очищается. Для регистров сведений, у которого в конфигураторе установлен режим записи "Подчинение регистратору", при записи значение регистратора всегда устанавливается той ссылкой, по которой был установлен отбор этого набора записей, независимо от того, что было назначено в процессе работы с набором

Следует отметить, что если вызвать метод "Записать" с параметром Истина для набора записей регистра сведений, у которого независимый режим записи, не установив перед этим отбор, то все записи регистра сведений будут удалены и замещены на те записи, которые мы добавили в набор. Записать набора записей регистра сведений, у которого режим записи "Подчинение регистратору", не установив предварительно отбор нельзя, и в данном случае возникнет ошибка.

Для работы с классами прикладных объектов в системе также предусмотрены предопределенные процедуры-обработчики, которые автоматически выполняются при наступлении различных событий. Например, процедуры: "ПриЗаписи", "ПередЗаписью", "ПриУдалении", "ОбработкаПроведения" и др. Такие процедуры располагаются в модулях объектов. При этом следует отметить, что такие предопределенные процедуры-обработчики вызываются на выполнение независимо от того, как было вызвано указанное событие, программно или интерактивно.

При выполнении некоторых предопределенных процедур, например, "ОбработкаПроведения", выполняется неявная транзакция. Для отмены неявной транзакции используется выражение:

```
Отказ = Истина;
```

Где Отказ - это параметр предопределенной процедуры

Задание 10

В документе "Приказ об изменении цен" создайте в процедуре "ОбработкаПроведения" движения по регистру сведений "ЦеныРеализации".

Отладчик

Отладчик - вспомогательный инструмент, облегчающий разработку программных модулей системы 1С:Предприятие 8.0. Отладчик предоставляет следующие возможности:

- Пошаговое выполнение модуля;
- Расстановка точек останова;
- Прерывание и продолжение выполнения модуля;
- Возможность отладки нескольких модулей одновременно;
- Вычисление выражений для анализа состояния переменных;
- Просмотр стека вызовов процедур и функций;
- Возможность остановки по возникновению ошибки;
- Возможность редактирования модуля в процессе отладки;
- Замер производительности.

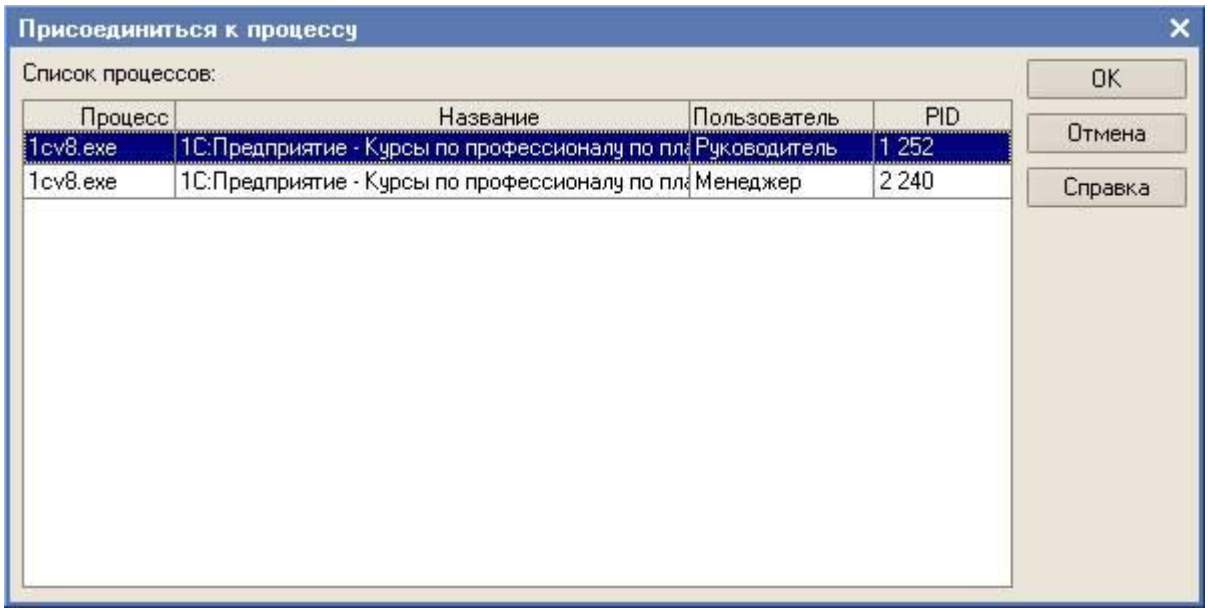
Отладчик является встроенным в конфигуратор инструментом. Для его использования необходимо, чтобы одновременно с конфигуратором была запущена отлаживаемая конфигурация в режиме "1С:Предприятие". Последовательность запуска программ не имеет значения.

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 "1С:Предприятие" для выбранной конфигурации, если режим уже был запущен, то выполните команду "Отладка - Подключиться";
- Выполните действия, которые вызовут исполнение отлаживаемого модуля (например, проведите документ);
- Проведите пошаговое выполнение нужного вам фрагмента модуля.

Если конфигурация запущена в режиме 1С: Предприятие, то необходимо в конфигураторе выбрать пункт "Отладка - Подключиться". На экран выводится окно для выбора процесса. Обычно список содержит одну строку с указанием на запущенную в режиме 1С: Предприятие конфигурацию. Если запущено несколько приложений 1С:Предприятие с данной конфигурацией, то список может содержать несколько строк. Выбор процесса указывает Отладчику, какой процесс будет отлаживаться.

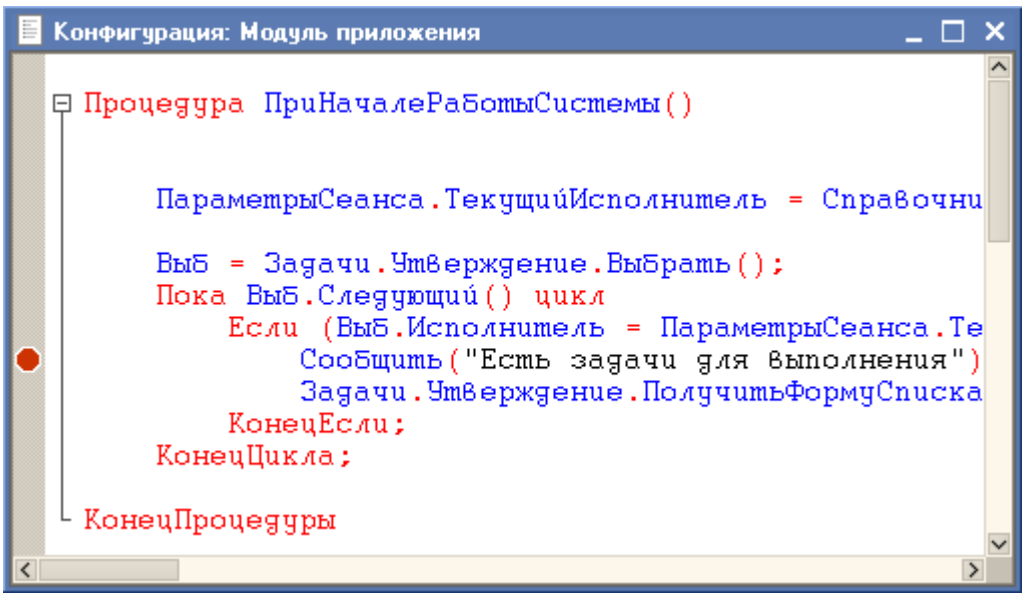


Для отладки модуля внешней обработки необходимо открыть файл внешней обработки в Конфигураторе, воспользовавшись пунктом "Файл - Открыть". В дальнейшем с модулем внешней обработки в Отладчике можно работать так же, как и с любым другим модулем.

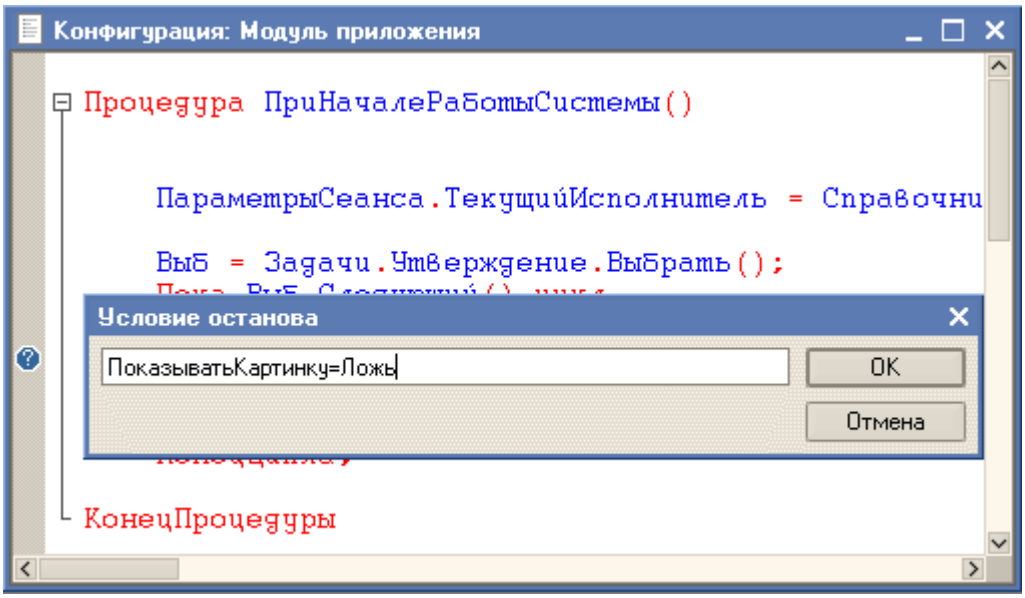
Отладчик позволяет установить на конкретную строку модуля специальный маркер - точку останова, - при достижении которой исполнение программного модуля останавливается и управление передается отладчику. Точку останова можно установить в любой строке модуля, в любой момент работы с Отладчиком. В случае, если строка, на которой устанавливается точка останова, не содержит операторов (например, пустая строка), содержит неисполняемый текст (например, заголовок процедуры или функции, определение переменных) или является продолжением оператора, начатого на предыдущих строках, положение точки останова будут автоматически скорректировано.

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

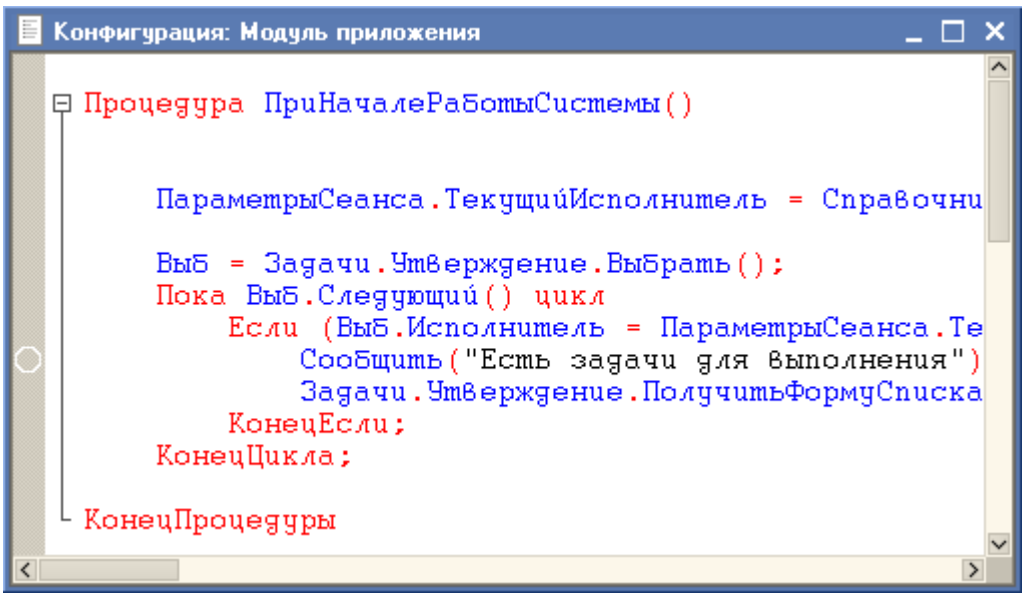
Точки останова могут быть безусловными или с условием. При достижении безусловной точки останова исполнение программного модуля останавливается в любом случае.



При достижении точки останова с условием, выполнение программного модуля останавливается только в том случае, если заданное условие истинно:

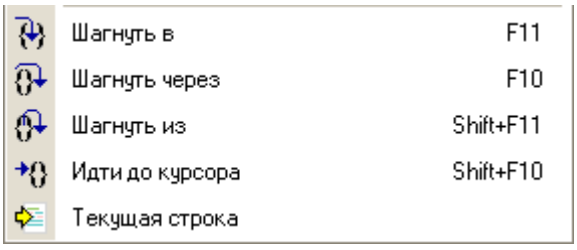


Отладчик поддерживает возможность отключения точек останова. При этом строка модуля остается отмечена маркером, однако на ход исполнения модуля он никакого влияния не оказывает.

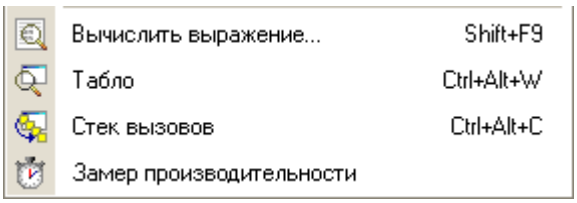


При большом количестве точек останова удобно использовать отдельное окно для работы с точками останова, позволяющее просматривать и редактировать их в едином списке:

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.



В случае, если не хотим продолжить пошаговое выполнение модуля дальше, с помощью команды "Отладка - Продолжить отладку" разрешим системе 1С:Предприятие 8.0 свободное выполнение модуля (до следующей точки останова).

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".

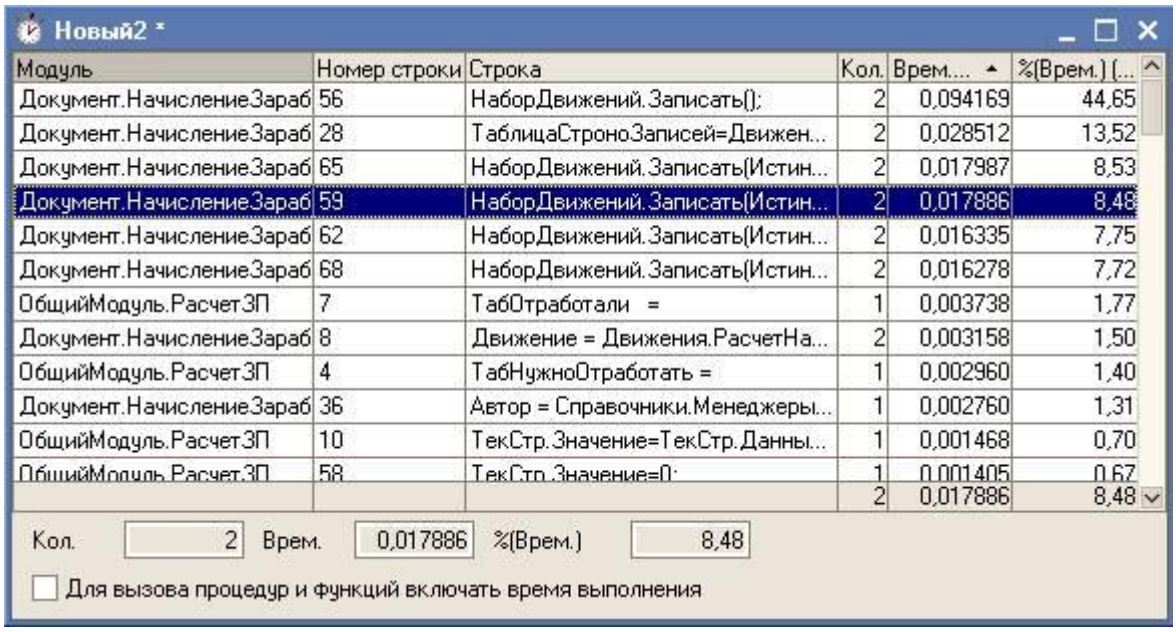
Установленные точки останова запоминаются при закрытии конфигурации. Если снова открыть конфигурацию, то посмотреть список точек останова можно с помощью диалога "Точки остановки", вызываемого пунктом меню "Отладка - Список точек останова".

В процессе отладки допускается редактирование текущей конфигурации и сохранение изменений.

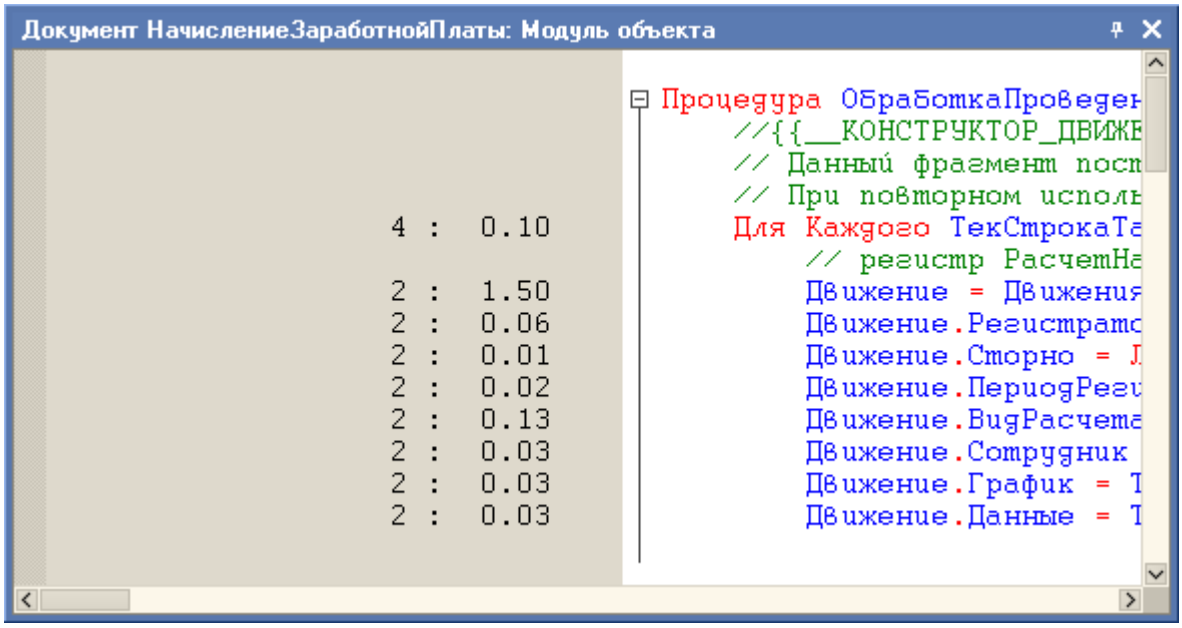
Внимание! Хотя в процессе отладки возможно редактирование отлаживаемого модуля, Отладчик не производит компилирование измененного кода - продолжается отладка кода конфигурации базы данных (на момент запуска отладчика или подключения). Для отладки изменений потребуется завершить работу в режиме "1С:Предприятие", сохранить изменения, обновить конфигурацию базы данных и повторно запустить Отладчик.

Режим замера производительности позволяет разработчику оценивать скорость работы как всей конфигурации в целом, так и отдельной ее части. В этом режиме измеряется частота использования конкретных участков кода и скорость их выполнения. Подобный анализ помогает выбирать наиболее оптимальный способ программной реализации алгоритмов работы системы, а также определять пути для повышения быстродействия прикладного решения.

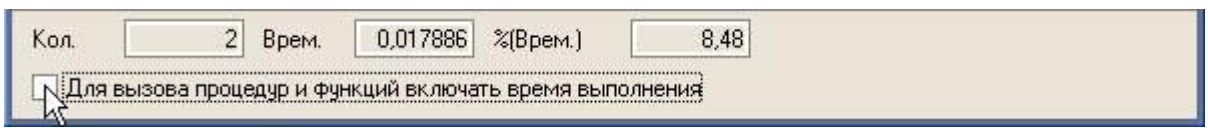
Результат замера производительности представляет собой список ссылок на конкретные строки модуля, с указанием частоты их выполнения и длительности (абсолютного времени выполнения и относительного, в процентах от общего времени выполнения замеряемого участка):



Результаты замера можно видеть также непосредственно в окне с исходным текстом модуля. Щелчком мыши на выбранной строке списка можно перейти к тексту модуля, для которого на отдельном поле будет отображено количество вызовов и относительное время выполнения каждой строки:



Результаты замера могут быть отсортированы по любой из колонок, например, по количеству вызовов строки. Кроме этого, режим замера производительности позволяет производить выборочное суммирование строк замера, для получения суммарных характеристик выполнения некоторых строк.



Дополнительной возможностью является включение в замер или исключение из замера времени выполнения вызываемых процедур и функций. Использование этой возможности

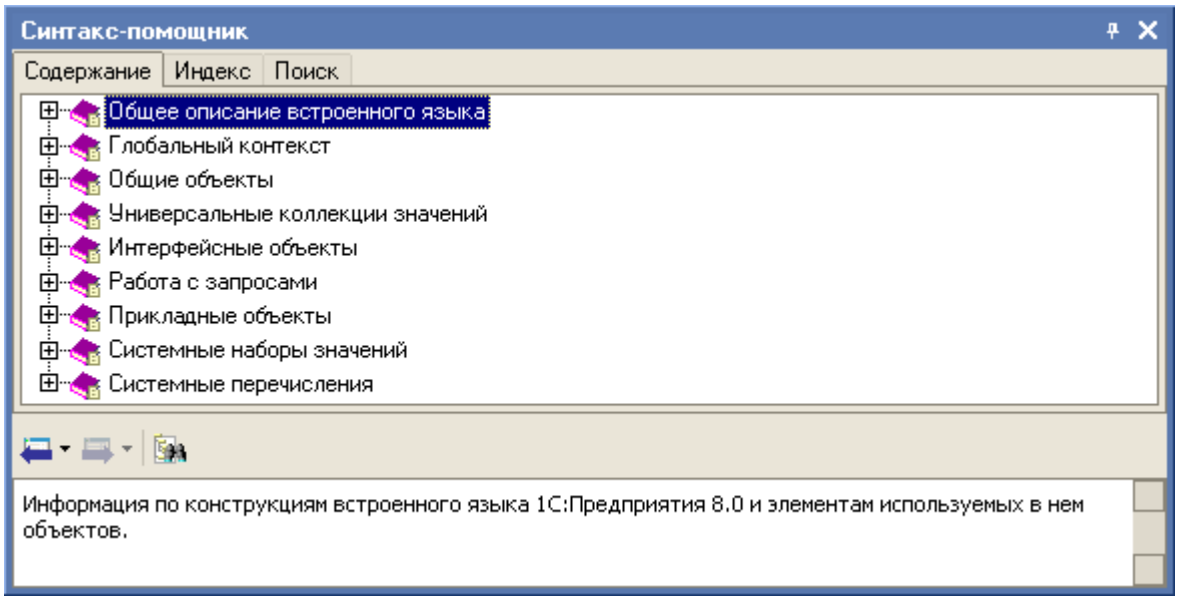
позволяет получать картину замера, максимально приближенную к реальной, в случае, когда из замеряемого модуля вызываются внешние по отношению к нему, процедуры.

Разработчик может сохранить результаты замера в файл для последующего анализа и сравнения с результатами других замеров.

Сервисные функции

Синтакс - помощник

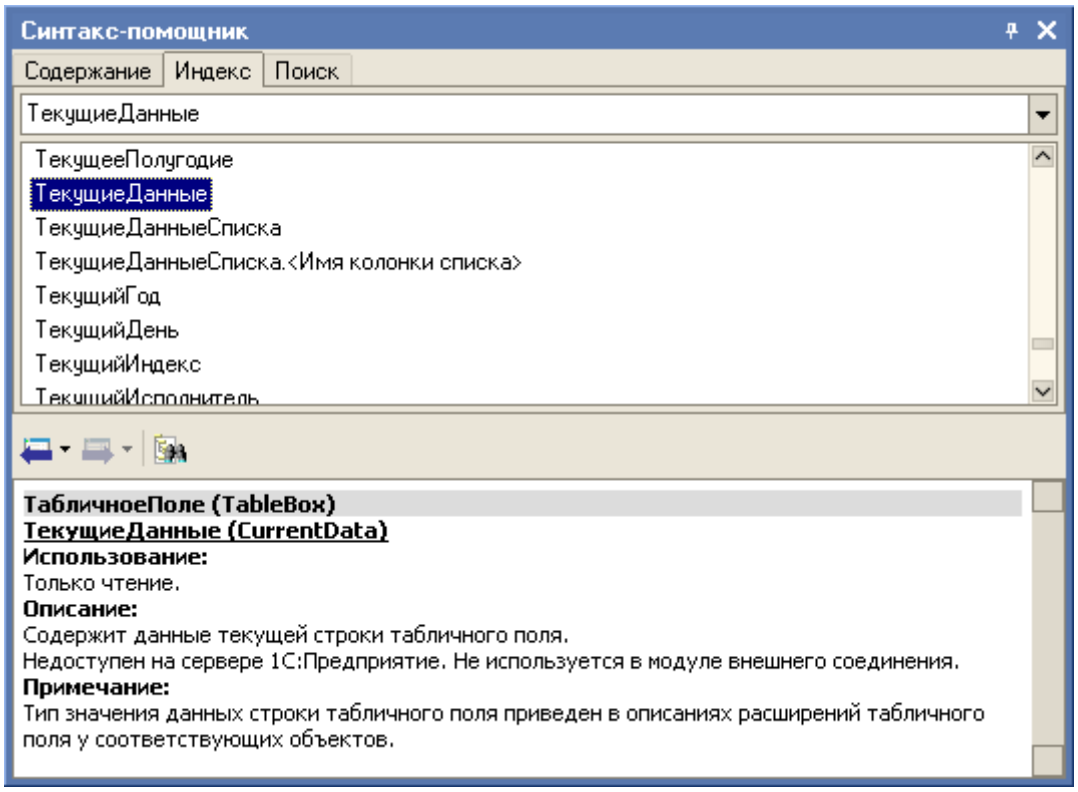
"Синтакс - Помощник" - средство, облегчающее разработку модулей. Основная задача "Синтакс - Помощника" - предоставить специалисту, выполняющему конфигурирование системы 1С:Предприятие 8.0, оперативную подсказку по встроенному языку. Для вызова "Синтакс - Помощника" в режиме "Конфигуратор" служит пункт "Справка - Синтакс - Помощник".



Синтакс-Помощник состоит из содержания, индекса, справочной информации. Содержание представляет структуру справочной информации в виде дерева и предназначено для быстрого перехода к нужной теме справки.

Индекс содержит список ключевых слов справки и используется для поиска по справочной информации.

Справочная информация отображает собственно страницу с информацией по выбранной теме/режиму приложения. Информация может быть представлена на английском и русском языках. Возможен поиск по подстроке и перетаскивание готовых синтаксических конструкций в модуль. Комбинация клавиш для вызова Синтакс-Помощника - Ctrl+Shift+F1.



Подсказку по конкретному элементу языка (оператору, процедуре, функции, свойству, методу) можно получить, если поместить курсор в модуле на этот элемент языка и нажать клавиши Ctrl+F1. В Синтакс - Помощнике будет выдано описание выбранного элемента встроенного языка.

Синтаксический контроль

Редактируемый модуль может быть проверен на правильность использования синтаксических конструкций встроенного языка. Для выполнения синтаксического контроля модуля необходимо воспользоваться пунктом "Текст - Синтаксический контроль".

Синтаксический контроль выполняется в следующей последовательности:

- Общие модули;
- Модуль приложения;
- Модуль объекта;
- Модуль формы.

При этом контроль модулей выполняется, если модуль еще не проходил контроля или был изменен. При контроле проверяются только те модули, которые в списке расположены до данного модуля.

При наличии ошибок их список будет выдан в окне сообщений с указанием полного адреса месторасположения и описания ошибки. Для перехода к строке модуля, вызвавшей ошибку, следует дважды щелкнуть мышью по этому сообщению.

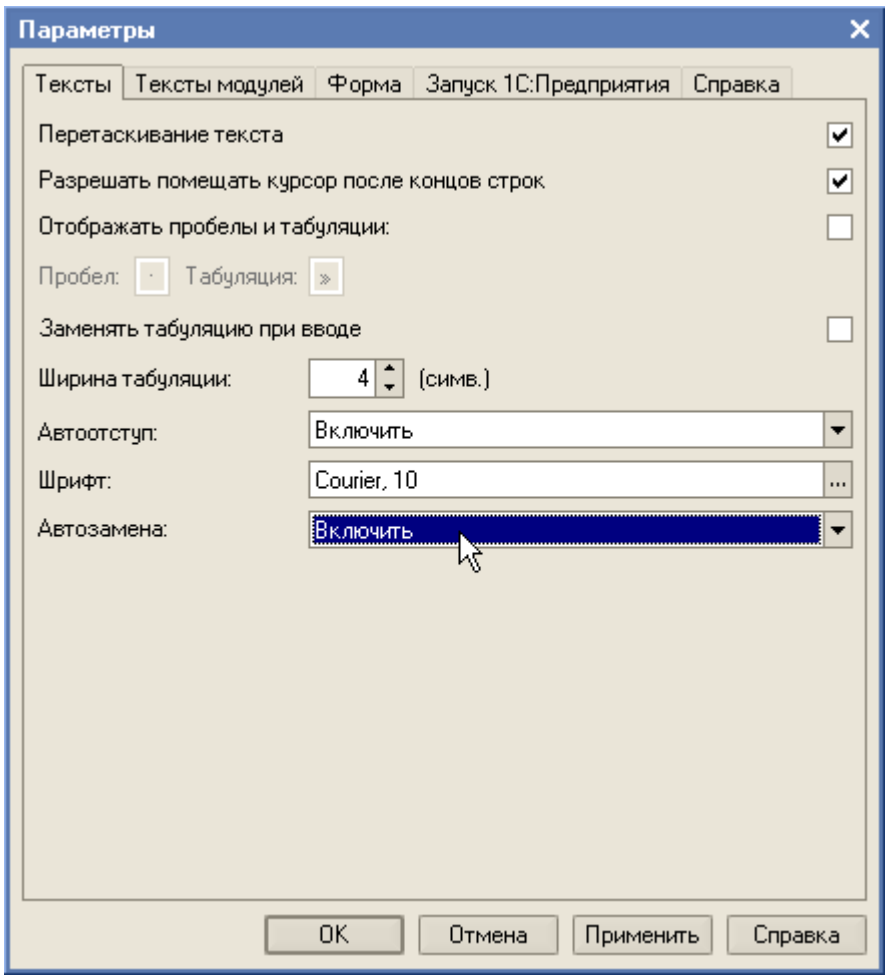
В режиме настройки параметров Конфигуратора (пункт "Сервис - Параметры" закладка "Текст модуля" реквизит "Проверять автоматически") можно включить режим автоматической проверки модуля. В этом случае, если модуль был изменен, при закрытии окна модуля или при сохранении конфигурации в целом будет выполняться синтаксический контроль модуля.

Для полного синтаксического контроля всех модулей конфигурации за один проход следует выбрать пункт "Конфигурация - Синтаксический контроль модулей".

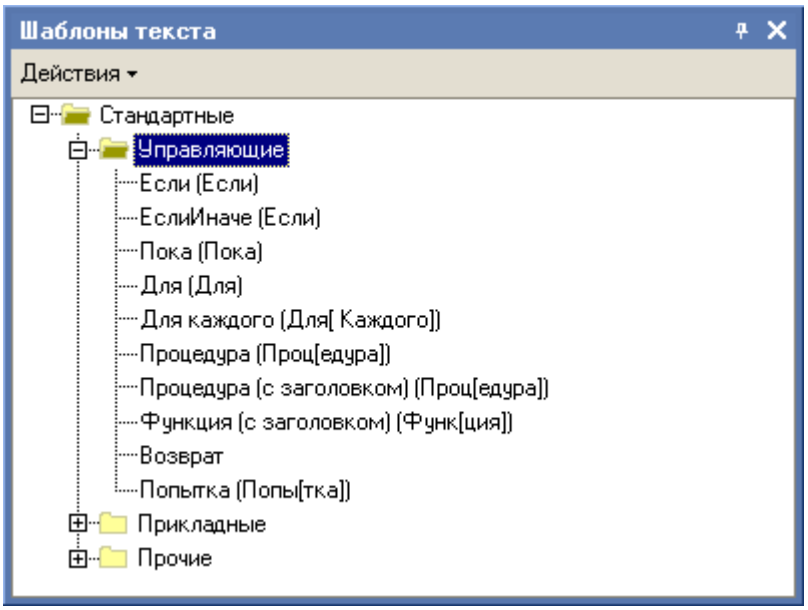
Использование шаблонов текста

Конфигуратор 1С:Предприятия 8.0 поддерживает возможность создания, сохранения и быстрой вставки часто используемых фрагментов текста. Такие фрагменты текста называются шаблонами.

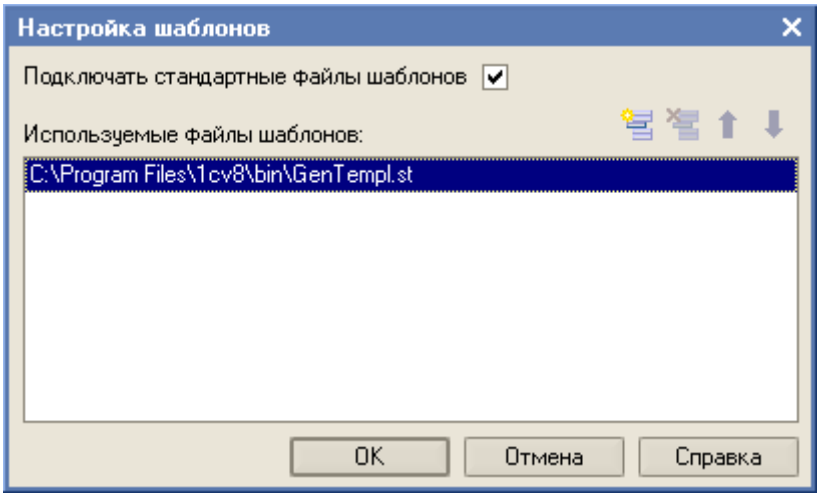
Если в параметрах Конфигуратора включен режим "автоподстановки", то нужный фрагмент текста будет автоматически вставляться в редактор при вводе шаблона (внимание: режим автоподстановки включается отдельно для модулей и текста на разных закладках окна настройки параметров Конфигуратора!).



Для управления списком шаблонов разработчик может использовать специальное окно, вызываемое из меню "Сервис - Шаблоны текста".

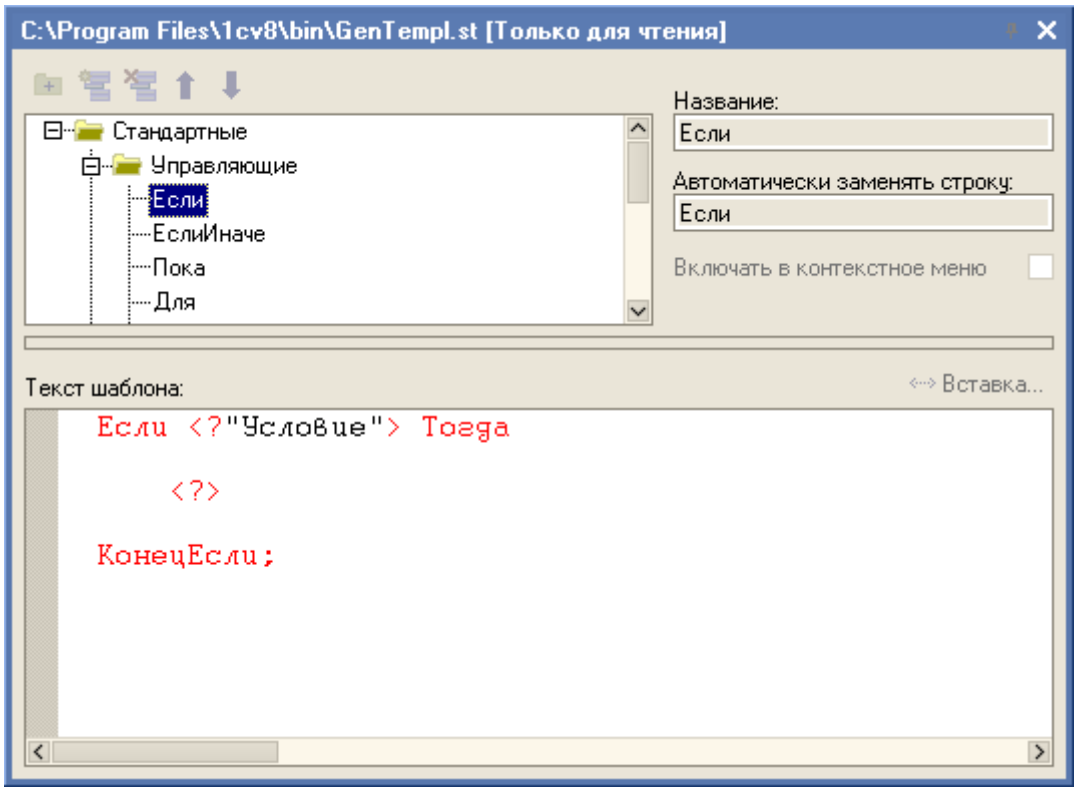


Все шаблоны хранятся в файлах с расширением *.st, и разработчик может создать новый файл шаблонов или подключить существующие через меню "Действия - Файлы шаблонов" из окна "Шаблоны текста".



Каждый шаблон представляет собой совокупность статической и динамической части. Статическая часть не изменяется и выводится в том виде, в котором она указана в шаблоне. Содержание динамической части зависит от контекста использования и может изменяться.

Разработчик имеет возможность создавать новые и редактировать существующие шаблоны. Редактирование шаблонов выполняется в специальном окне, которое вызывается через меню "Действия - Изменить" окна "Шаблоны текста" и содержит список шаблонов и текст выбранного шаблона. Динамическая часть шаблона выделяется специальными символами < >.



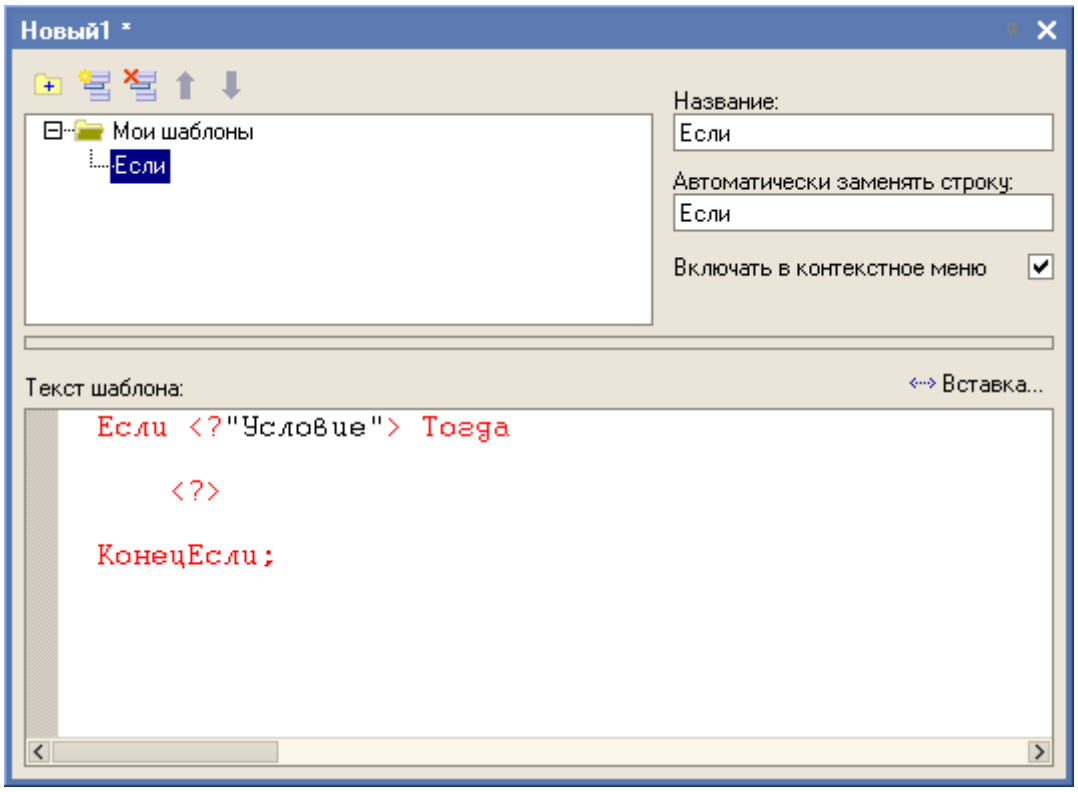
При редактировании текстового документа или модуля, возможна быстрая вставка шаблона после того, как будет набрана последовательность символов, указанная в шаблоне. Вставка производится при нажатии клавиши Пробел или Enter.

Кроме этого существует возможность непосредственной вставки любого шаблона путем выбора его из контекстного меню, для этого необходимо установить флажок "Включать в контекстное меню".

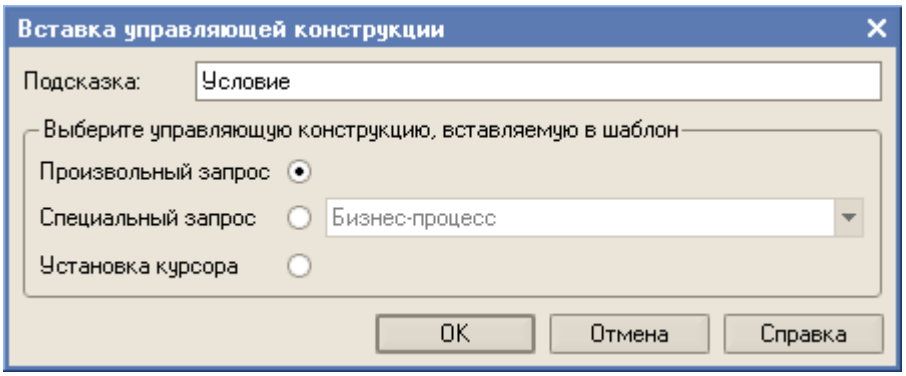
Для универсальности использования шаблона применяется механизм вставки управляющей конструкции. В тексте шаблона размещают управляющие конструкции - последовательности символов, которые при вставке заменяющего текста шаблона вызывают выполнение каких - либо действий. Например, управляющие конструкции позволяют запрашивать у пользователя какую - либо информацию и помещают эту информацию во вставляемый текст.

Управляющие конструкции в шаблон можно вставлять вручную или использовать специальный запрос для интерактивной вставки управляющих конструкций.

Для вызова диалога "Вставка управляющей конструкции" необходимо в окне редактирования шаблона нажать кнопку "<->Вставка".



Группа переключателей в окне "Вставка управляющей конструкции" позволяет выбрать вид управляющей конструкции, вставляемой в шаблон.



Произвольный запрос. Используется для ввода произвольного текста. В тексте шаблона появится управляющая конструкция вида

- **Остатки()** - получает остатки регистра накопления на заданный момент времени. Есть возможность фильтрации по значениям измерений, а также получения остатков в разрезе других измерений. Возвращает таблицу значений, содержащую колонки с измерениями, указанными в параметре Измерения, и колонки с ресурсами, указанными в параметре Ресурсы. Имеет смысл только для регистров, у которых в конфигураторе указан вид регистра "Остатки".
- **ПолучитьФорму()** - получает форму регистра накопления.
- **СоздатьНаборЗаписей()** - создает набор записей регистра накопления. Набор записей создается пустым. Возвращаемое значение имеет тип "РегистрНакопленияНаборЗаписей".

Пример получения остатка конкретного материала из регистра накопления "ОстаткиМатериалов":

```
ТекМатериалОтбор = Новый Структура("Материал",ВыбМатериал);

ТабЗначенийМатериалОстатки = РегистрыНакопления.ОстаткиМатериалов.Остатки(Дата,
ТекМатериалОтбор, "Материал ","Количество,Сумма");

Сообщить(ТабЗначенийМатериалОстатки[0].Количество);
```

Основной способ добавления и изменения записей регистра накопления, как и других регистров тоже, - через набор записей регистра.

У документов есть свойство "Движения", тип - ФиксированнаяКоллекция, которое предоставляет доступ к коллекции наборов записей движений документа. Свойства коллекции содержат пустые наборы записей движений документа, которые включены для данного документа в конфигурации. Поэтому при проведении документов сначала добавляются записи в набор, а затем набор записывается в базу данных.

Через набор записей также можно обращаться к уже существующим записям регистра накопления. Для этого нужно установить свойство Отбор и прочитать записи из базы данных. Свойство Отбор является объектом типа Отбор, свойства которого совпадают с именами измерений регистра и являются объектами типа ЭлементОтбора.

Например:

```
Набор = РегистрыНакопления.Услуги.СоздатьНаборЗаписей();
Набор.Отбор.Регистратор.Значение = ВыбДок;
Набор.Прочитать();
Для Каждого Движ из Набор цикл
Сообщить(Движ.Сумма);
КонецЦикла;
```

Движения по регистрам накопления в основном создаются при проведении документов. Для того чтобы документ мог делать движения по регистру, необходимо на закладке "Движения" указать следующие параметры:

Проведение - разрешает или запрещает проведение документа при записи

Оперативное проведение - разрешает или запрещает оперативное проведение. Если оперативное проведение разрешено, то система позволяет при проведении документа текущей датой интерактивно выбирать метод проведения документа. Для документов с разрешенным оперативным проведением при выборе не текущей даты проведение осуществляется в неоперативном режиме, так как учитывается уже свершившийся факт, который не требует контроля, осуществляемого в оперативном режиме, например, проверка остатка, указанного в расходной накладной товара. Для того чтобы проводить документы будущей датой необходимо запретить Оперативное проведение.

Удаление движений - задает автоматическое удаление всех записей при перепроведении или отмене проведения документа, которые документ записал в процессе предыдущего проведения

Также необходимо отметить те регистры накопления, по которым данный документ будет делать движения.