**Вычисление логических выражений в операторах встроенного языка**

В условных операторах и операторах цикла встроенного языка системы 1С:Предприятие для определения последовательности выполняемых действий используются логические выражения. Логическим выражением может быть элементарное логическое выражение (сравнение двух значений) или более сложные конструкции полученные из элементарных логических выражений при помощи логических операций И/ИЛИ/НЕ.

Логические выражения могут быть весьма сложными и изощренными. И здесь у разработчика конфигурации может возникнуть сложности, связанные с неправильным пониманием правил вычисления логических выражений. Рассмотрим пример.

Предположим, имеется процедура, предназначенная для обработки некоторого реквизита справочника или документа. Тип данного реквизита не определен, а процедура может вызываться для любого значения реквизита вне зависимости от его типа. Процедура может иметь вид:

Процедура Обработать(Об)

Если (ТипЗначенияСтр(Об) = "Справочник") И (Об.Вид() = "Товары") Тогда

// Некоторые действия

КонецЕсли;

...

КонецПроцедуры

Если в качестве значения параметра, передаваемого в данную процедуру, будет фигурировать элемент справочника любого вида, то процедура будет работать нормально, но если в качестве параметра будет передано значение типа "Число" или "Дата", то выполнение процедуры будет аварийно завершено. В логическом выражении, используемом в операторе 'Если' только составляющая ТипЗначенияСтр(Об) = "Справочник" будет корректно функционировать независимо от типа передаваемого параметра, а составляющая Об.Вид() = "Товары" сможет нормально работать только в том случае, если в качестве параметра в процедуру передано значение агрегатного типа и для данного агрегатного типа определен метод Вид(). Дело в том, что при вычислении логических выражений встроенный язык системы 1С:Предприятие 7.х сначала вычисляет элементарные логические выражения (операции сравнения), а затем производит вычисление логического выражения в целом (операции И/ИЛИ/НЕ).

Таким образом, данная процедура не сможет корректно работать с параметром заранее не определенного типа и может при выполнении вызвать ошибку, что неприемлемо. Очевидно, что для правильного функционирования процедуры при ее разработке следовало учесть порядок вычисления логических выражений, принятый во встроенном языке системы 1С:Предприятие 7.х. То есть, чтобы работоспособность процедуры сохранялась при любом значении параметра, процедура должна выглядеть как:

Процедура Обработать(Об)

Если ТипЗначенияСтр(Об) = "Справочник" Тогда

Если Об.Вид() = "Товары" Тогда

// Некоторые действия

КонецЕсли

КонецЕсли

...

КонецПроцедуры

В данном случае порядок вычисления логических выражений в операторах Если выражен в явном виде и выражение Об.Вид() = "Товары" будет вычисляться только в том случае если условие ТипЗначенияСтр(Об) = "Справочник" окажется истинным и ошибки при выполнении процедуры не произойдет при любом значении параметра.

Заметим, что приведенный выше порядок вычислений не распространяется на вычисление выражений определяющих результат оператора "?". Само условие оператора "?" вычисляется по общим правилам логических выражений описанным выше, но из двух выражений определяющих результат оператора "?" выполняется только одно (в зависимости от результата логического выражения). То есть, в одном из выражений определяющих результат оператора "?" (не вычисляемом) ошибок времени выполнения не будет возникать вне зависимости от используемых в нем конструкций.

Особенности применения предопределенных процедур встроенного языка

Во встроенном языке системы 1С:Предприятие 7.x предусмотрен ряд системных предопределенных процедур При..., таких как "ПриЗаписи", "ПриУдаленииДокумента", "ПриУдаленииЭлемента", "ПриЗаписиКонстанты", "ПриОтменеПроведенияДокумента" и т. п., определяемых в различных модулях конфигурации и вызываемых системой при выполнении соответствующих действий. Назначение указанных процедур состоит в том, чтобы специалист по конфигурированию мог предусмотреть необходимые проверки при интерактивном выполнении соответствующих операций, и, в случае необходимости, имел возможность запретить их выполнение.

При этом следует учитывать следующие особенности данных процедур:

1. Указанные предопределенные процедуры не связаны непосредственно с действиями, при выполнении которых они вызываются, а сам вызов производится только при интерактивном выполнении указанных действий. Например, процедура "ПриУдаленииДокумента" будет вызвана в том случае, когда удаление документа будет инициировано "нажатием кнопки" в форме журнала. В то же время вызов этой предопределенной процедуры не будет производиться, если удаление документа производится из какого-либо модуля встроенного языка (например, из некоторой обработки).

2. Вызовы указанных предопределенных процедур не объединены с выполняемым действием единой транзакцией. Это означает, что в случае интерактивного удаления документа, само удаление документа выполняется в транзакции, отдельной от действий, которые могут выполняться в процедуре "ПриУдаленииДокумента". И если в процедуре "ПриУдаленииДокумента" действия будут выполняться в раках одной, явно определенной в процедуре транзакции, то все равно приходится говорить о двух транзакциях:

- удаление документа;

- действия, выполняемые в процедуре "ПриУдаленииДокумента"

Примером неправильного применения таких предопределенных процедур является использование предопределенных процедур группы "При..." для выполнения действий критичных с точки зрения поддержки целостности данных. Например, при записи документа создается, записывается и проводится ряд связанных с ним документов (процедура "ПриЗаписи" модуля формы документа). И эти же связанные документы удаляются при удалении "ведущего" документа (процедура "ПриУдаленииДокумента" глобального модуля).

Такую технику вряд ли следует признать правильной, в силу приведенных выше особенностей предопределенных процедур. Так как предопределенные процедуры не связаны жестко с выполняемыми действиями, то вполне вероятно, что при очередном изменении конфигурации удаление документа станет выполняться не из формы журнала, а из специально созданной обработки, что приведет к тому, что процедура "ПриУдаленииДокумента" в данном случае вызываться не будет и, соответственно, не будут произведены необходимые для этого случая действия. Но если возникновение ситуаций такого рода еще можно как-топредотвратить путем более тщательной проработки конфигурации, то объединить действия, выполняемые в рамках процедуры "ПриУдаленииДокумента" и собственно удаление документа в одну транзакцию не возможно. При этом разнесение единого по логике действия в разные транзакции может быть весьма опасно с точки зрения поддержания логической целостности данных, так как ситуация, когда одно действие выполняется, а другое по каким-либо причинам - нет, вполне вероятна, особенно в многопользовательских системах.

Заметим, что описанные выше особенности предопределенных процедур не распространяются на предопределенные процедуры "ОбработкаПроведения" и "ОбработкаУдаленияПроведения". Эти предопределенные процедуры вызываются непосредственно в транзакции проведения (отмены проведения) и выполняемые в них алгоритмы логически связанны с самим выполняемым действием. Кроме того, в отличие от других предопределенных процедур эти предопределенные процедуры выполняются и в случае проведения (отмены проведения) средствами встроенного языка.

**Сохранение (восстановление) значений в строку**

В 1С:Предприятии существует несколько способов, позволяющих преобразовать значение к строковому виду и восстановить его обратно из строки.

Заметим, что в данном случае речь идет не о преобразовании к строке значения вызовом функции "Строка" или аналогичном автоматическом преобразовании в выражении строкового типа. Такое преобразование используется лишь для получения визуального представления значения и кроме типов "число", "строка", "дата", не имеет возможности обратного преобразования.

В данном разделе описываются способы, которые ориентированы на преобразование значения в строку с целью его обратного преобразования к исходному типу. Эти возможности используются обычно для решения специальных задач, связанных с хранением значений вне информационной базы или передачи данных в другую информационную базу или программу.

Следует учитывать, что применение таких преобразований требует хорошего понимания данного механизма и системы 1С:Предприятия в целом.

Какие значения преобразуются в строку.

К строковому виду могут быть преобразованы значения любых типов, которые могут храниться в информационной базе, то есть тех типов, которые доступны, например, при создании реквизита документа. Это, прежде всего, базовые типы ("число", "строка", "дата"), а также типы определяемые метаданными ("перечисление", "счет", "справочник", "документ" и т.д.).

Кроме того, к строке могут быть преобразованы значения типа "СписокЗначений" и "ТаблицаЗначений", которые фактически представляют собой набор отдельных значений.

Значения других типов используемых во встроенном языке 1С:Предприятия, но не хранящиеся в информационной базе (например, "Таблица" или "Запрос") не могут быть преобразованы к строке.

При сохранении в строку значений типа "СписокЗначений" и "ТаблицаЗначений" допускается вложенное преобразование значений. То есть при преобразовании списка значений будет преобразовываться в строку все значения входящие в список, включая списки значений и таблицы значений. Разумеется, не будут преобразовываться в строку входящие в список значения, которые сами не могут быть преобразованы к строке.

Также следует иметь в виду, что преобразуются к строке отдельные значения, а не объекты данных. То есть, можно преобразовать к строке значение реквизита элемента справочника, но нельзя преобразовать к строке весь элемент справочника целиком. Для сохранения в строке элемента справочника, необходимо сохранять отдельно все его реквизиты.

Способы преобразования к строке.

Значения могут преобразовываться к строке в двух разных форматах - внутреннем и внешнем. Использование этих форматов существенно отличается по своему назначению и возможностям. При преобразовании к строке формат (внутренний или внешний) всегда задается явно. Он определяется именем функции или ее параметром. При обратном преобразовании (из строки) необходимо вызывать соответствующую функцию преобразования именно из того формата, в котором выполнялось сохранение значения в сроковом виде.

Внутренний формат.

Для преобразования к строковому виду (и обратно) во внутреннем формате используются следующие функции:

- ЗначениеВСтрокуВнутр / ЗначениеИзСтрокиВнутр;

- ЗначениеВФайл / ЗначениеИзФайла - параметр "Формат" =1;

- СохранитьЗначение / ВосстановитьЗначение.

Кроме того, во внутреннем строковом формате хранятся значения расшифровок табличного документа.

Значение, полученное в сроковом виде во внутреннем формате, может быть преобразовано из строки к исходному типу только в той же информационной базе. Это объясняется тем, что для сохранения значения во внутреннем формате используются системные идентификаторы (не отображаемые на экране), как объектов метаданных, так и объектов данных. При попытке преобразовать значение, сохраненное во внутреннем формате, в другой конфигурации может произойти неправильное преобразование значения.

Если объект данных в информационной базе, на который ссылалось значение, удален (например, элемент справочника), то при преобразовании значения из строки обращение к этому значению не вызовет системной ошибки, но, разумеется и не позволит получить данные удаленного объекта.

Строки, полученные преобразованием значений к строке во внутреннем формате нельзя хранить в информационной базе. Прежде всего, это приведет к нарушению работы механизма контроля ссылочной целостности. Также, это может повлиять на работу компоненты "Управление распределенными данными ИБ". Разумеется, такое хранение значений, также не позволит системе оптимизировать обращение к данным при использовании версий 1С:Предприятия для SQL. Заметим, что хранение в информационной базе значений преобразованных в строку нарушает идеологию 1С:Предприятия и затрудняет отладку и модификацию конфигураций.

Отдельно следует отметить возможность использования преобразования значений к строке во внутреннем формате для однозначной идентификации объектов, при переносе информации из одной информационной базы в другую. То есть, при загрузке некоторых данных из другой информационной базы строковые представления значений другой ИБ в данной ИБ не могут использоваться собственно для обратного преобразования, но могут использоваться для сопоставления двух значений той ИБ, из которой идет загрузка. Например, если загружается справочник, то ссылку на родителя загружаемого элемента можно сопоставить со ссылкой ранее загруженной группы справочника, для точной идентификации родителя. Разумеется, сопоставлять можно только строки, полученные в одной информационной базе. То есть в описанном примере строковое представление во внутреннем формате должно выгружаться и для самих групп и для родителей элементов, чтобы при загрузке их можно было сопоставить. Для сопоставления можно рекомендовать запоминать при загрузке групп строковые представления этих групп в исходной ИБ, а при загрузке элементов искать группу соответствующую загружаемому элементу по строковому представлению родителя элемента в исходной ИБ.

Внешний формат

Для преобразования к строковому виду во внешнем формате используются

следующие функции:

- ЗначениеВСтроку / ЗначениеИзСтроки;

- ЗначениеВФайл / ЗначениеИзФайла - параметр "Формат" =0.

В отличие от внутреннего формата, при преобразовании значения в строку во внешнем формате система использует не системные идентификаторы объектов метаданных и данных, а обычные строковые идентификаторы объектов метаданных и те значения данных, на основании которых можно однозначно идентифицировать объект. Например, для преобразования к строке во внешнем формате значения типа "Справочник.Товары" будет использован идентификатор справочника "Товары", а также код элемента справочника. Если для кода справочника уникальность установлена в пределах группы, то будет использован полный код с учетом всех вышестоящих групп. Если справочник является подчиненным, то для идентификации значения будут использован код элемента справочника - владельца. Разумеется, если для справочника отключен контроль уникальности кодов, то преобразование значения этого типа практически не имеет смысла. Для сохранения значения типа "документ", если уникальность номеров поддерживается в пределах некоторого интервала (года, месяца), то для сохранения и восстановления значения используется также дата документа.

Внешний формат преобразования к строке может использоваться для переноса значений между различными информационными базами. Разумеется, успех обратного преобразования зависит от того, имеются ли в той информационной базе, в которой значение преобразуется из строки объекты метаданных с соответствующими идентификаторами и объекты данных соответствующими кодами/номерами. Например, полученное нами в строковом виде значение типа "Справочник.Товары", будет правильно обратно преобразовано, если имеется справочник, товары и элемент справочника с тем кодом, который был у элемента, на который ссылалось значение преобразованное к строке.

При восстановлении значения из строки во внешнем формате преобразование может быть не выполнено по нескольким причинам. Например, в той конфигурации, в которой происходит преобразование из сроки, не существует объекта метаданных с идентификатором, соответствующим типу значения, сохраненного в строке. Другой причиной может быть отсутствие в информационной базе объекта данных с тем кодом (номером), который был у элемента справочника, документа или счета, на который ссылалось сохраненное в строке значение. Кроме того, преобразование может быть не выполнено из-за отличий в свойствах объектов метаданных. Например, если при сохранении в строку значения типа "Справочник.Товры" у справочника "Товары" была установлена уникальность кода в пределах подчинения, а при восстановления значения из строки уникальность кода была установлена в пределах всего справочника, то значение из строки не будет восстановлено. Однако, неудачное преобразование из строки не вызывает системной ошибки, а приводит лишь к получению пустого значения.

Заметим, что возможность преобразования к строке во внешнем формате может использоваться при передаче данных из одной информационной базы 1С:Предприятия в другую с помощью Ole Automation, так как непосредственно передать значение типа "справочник", "документ" и т.п. через Ole

Automation нельзя.

Следует учитывать, что восстановление значений типа "справочник", "документ" или "счет" из строки во внешнем формате в отличие от использования внутреннего формата занимает определенное время, так как при этом выполняется обращение к информационной базе для поиска значений по сохраненному номеру (коду).

Заметим, что преобразование к строке и обратно значений типа "число", "строка" и "дата" происходит однозначно (гарантированно) в обоих форматах, так как эти значения не зависят от конфигурации. В качестве примера такого сохранения в методической конфигурации в обработке "НастройкаПрописи" выполняется сохранение в файле строк, используемых для вывода чисел, дат и периодов прописью.

**Работа со значениями агрегатных типов во встроенном языке**

При работе со значениями агрегатных типов во встроенном языке системы 1С:Предприятие следует учитывать некоторые особенности, которые не всегда являются очевидными.

Каждое значение агрегатного типа в системе 1С:Предприятие включает две составляющие:

внутренний идентификатор объекта данных;

ссылка на объект доступа.

Внутренний идентификатор объекта предназначен для максимально полной идентификации объекта, соответствующего значению агрегатного типа. А объект доступа служит для обращения к объекту данных посредством реквизитов и методов. При выполнении присваивания переменной встроенного языка значения агрегатного типа, в переменную копируется внутренний идентификатор объекта и ссылка на объект доступа, если таковой имеется. Сам объект доступа при этом не копируется! Таким образом, в результате

выполнения оператора

A = B;

при условии, что переменная B имела значение агрегатного типа и уже имела ссылку на объект доступа, переменные A и B будут ссылаться на один и тот же объект доступа. В отличие от базовых типов (число, строка, дата), для которых при присваивании копируется само значение, при присваивании агрегатных типов копируется не объект доступа, а только ссылка на него.

Здесь, однако, следует учитывать, что не всегда значение агрегатного типа содержит ссылку на тот или иной объект доступа. В некоторых случаях объекта доступа, соответствующего значению агрегатного типа может еще не существовать.

Значение агрегатного типа всегда содержит ссылку на объект доступа в тех случаях, когда значение было создано с помощью функции СоздатьОбъект(). А вот у атрибутов запросов, реквизитов справочников, документов и т. п., имеющих агрегатный тип, значения изначально не содержат ссылок на объекты доступа, пока они не будут созданы при обращении к атрибутам и методам. Соответственно и при присваивании таких значений переменным встроенного языка, переменные также не приобретают ссылок на объекты доступа, если они еще не были созданы.

Как же тогда в таких случаях происходит обращение к атрибутам и методам? Дело в том, что значение агрегатного типа, у которого нет ссылки на объект доступа может существовать только для конкретных типов системы 1С:Предприятие - элементов справочников, документов, счетов бухгалтерского учета и т. п.. И здесь выручает внутренний идентификатор объекта, который всегда содержится в значении агрегатного типа и для указанных типов может предоставить исчерпывающую информацию, необходимую для создания объекта. А само создание объекта доступа производится при первом обращении его методу или атрибуту. Поясним вышесказанное на примере:

Тов = Док.Товар; // Объект доступа еще не создан, при условии, что ранее не было обращений к атрибутам и методам;

Тов2 = Док.Товар; // Аналогично

Цена = Тов.Цена; // Для переменной Тов создается объект доступа

Цена2 = Тов2.Цена; // Для переменной Тов2 создается объект доступа. Тов и Тов2 ссылаются на разные объекты доступа

Тов3 = Тов; // Тов и Тов3 ссылаются на один объект доступа

При создании объекта доступа происходит считывание данных объекта информационной базы. В дальнейшем значения прочитанных реквизитов остаются неизменными, вне зависимости от изменений, производимых в информационной базе. Поэтому не следует, сохранять в переменных на продолжительный период значения, соответствующие элементам справочников, документам и т. п., в надежде, что в их реквизитах будут отображаться актуальные на текущий момент значения.

Надо отметить, что объекты доступа продолжают существовать до тех пор, пока на них есть ссылки. Для приведенного выше примера уничтожение объектов, созданных для доступа к реквизитам элемента справочника может быть вызвано выполнением последовательности операторов:

Тов = 0; // Удалена ссылка на объект доступа

Тов2 = 0; // Уничтожен первый объект доступа

Тов3 = 0; // Уничтожен второй объект доступа

В результате выполнения этих операторов ссылки на объекты доступа удаляются путем присваивания переменным других значений. Ссылки на объекты доступа были бы также уничтожены при выходе из процедуры или функции, для которой переменные Тов, Тов2 и Тов3 были локальными. Если же переменная, содержащая ссылку на объект доступа, является глобальной переменной глобального модуля, то единственный способ уничтожить ссылку на объект доступа - это присвоить переменной другое значение.

Теперь рассмотрим случай, когда обращение к реквизиту происходит не через одну, как в рассмотренных выше примерах, а через две и более точек? Например, в случае выполнения оператора:

Цена = Док.Товар.Цена;

В этом случае, исполняющая система встроенного языка получает значение реквизита Товар документа, представленного значением переменной Док. При этом полученное значение не будет содержать ссылки на объект доступа. Затем, исполняющая система создает объект доступа, соответствующий реквизиту Товар, и, затем, с помощью созданного объекта, получает значение реквизита Цена. Затем объект доступа удаляется, так как ни одна из переменных после выполнения указанного оператора на объект не ссылается. Таким образом, при выполнении последовательности операторов

Наимен = Док.Товар.Наименование;

Цена = Док.Товар.Цена;

объект доступа, соответствующий Док.Товар будет дважды создан и дважды уничтожен. С точки зрения производительности для достижения той же цели было бы эффективнее воспользоваться последовательностью операторов:

Тов = Док.Товар; // Присваивается реквизит документа. Объект доступа еще не создан

Наимен = Тов.Наименование; // Для переменной Тов создается объект доступа

Цена = Тов.Цена; // Используется созданный ранее объект доступа

При выполнении данной последовательности объект для доступа к элементу справочника создается однократно и, после выполнения указанных операторов, переменная Тов сохранит ссылку на объект доступа и он сможет быть использован в дальнейшем, до тех пор, пока переменной не будет присвоено другое значение или произойдет возврат из процедуры или функции, в которой переменная определена.

**Значения типов, соответствующих объектам информационной базы во встроенном языке**

Для таких объектов информационной базы системы 1С:Предприятие как справочники, документы и счета бухгалтерского учета значения, представляющие эти объекты во встроенном языке могут выступать в двух разных видах:

- значения, полученные путем вызова системной функции СоздатьОбъект

- значения полученные из реквизитов форм и атрибутов объектов информационной базы

Значения, полученные при помощи вызова СоздатьОбъект() не представляют какой-то конкретный объект (элемент справочника, документ, счет бухгалтерского учета), а предназначены главным образом для обработки групп объектов. С помощью таких значений можно получать выборки множества объектов, последовательно обходить объекты, попавшие в выборку, производить их обновление, создавать новые и удалять существующие объекты и т. п.. Такие значения могут использоваться "многократно", то есть с помощью одного значения можно получить доступ сначала к одной выборке, потом к другой и так много раз.

Значения, содержащихся в реквизитах форм и атрибутах объектов информационной базы, являются точными значениями и представляют ровно один конкретный объект информационной базы (элемент справочника, документ, счет бухгалтерского учета). Такие значения также могут быть пустыми – не представляющими никакого объекта. С помощью таких значений можно получать доступ к атрибутам соответствующего объекта, но нельзя производить действий по созданию, удалению и изменению объекта.

Но несмотря на такие существенные различия оба описанных варианта значений одинаковы во многих проявлениях. Рассмотрим пример:

Спр1 = СоздатьОбъект("Справочник.Товары");

Спр2 = Док.Товар;

Переменные Спр1 и Спр2 содержат значения двух описанных выше видов. Для этих переменных способ доступа к атрибутам справочника производится одним и тем же способом. Многие методы объекта типа Справочник могут применяться как к Спр1, так и Спр2. Функции ТипЗначения() и ТипЗначенияСтр(), будучи примененными к Спр1 и Спр2 дадут одинаковые результаты. Многие методы встроенного языка также не различают какого вида значение передано им в качестве параметра: точное значение объекта или спозиционированная выборка.

И такое положение дел приводит к тому, что иногда у специалистов по конфигурированию возникают трудности с пониманием различия между приведенными видами значений. В результате при написании конфигураций возникают ситуации, когда одной и той же процедуре в качестве параметра могут передаваться значения разных видов:

1.

Спр1 = СоздатьОбъект("Справочник.Товары");

Спр1.ВыбратьЭлементы();

Пока Спр1.ПолучитьЭлемент() = 1 Цикл

Обработать(Спр1);

КонецЦикла;

2.

Спр2 = Док.Товар;

Обработать(Спр2);

В первом случае в качестве параметра процедуре Обработать() передается выборка, спозиционированная на конкретный элемент справочника, а во втором - точное значение элемента справочника. И, если, например, в процедуре Обработать() производится только считывание реквизитов справочника, то результаты, полученные в обоих случаях будут неразличимы. Но если в процедуре Обработать() производится присвоение переданного в качестве параметра значения реквизиту формы или атрибуту объекта (например, документа), то первый вариант просто не будет работать, так как вместо точного значения элемента справочника процедура получит выборку. Для того, чтобы все работало нормально первый вариант должен выглядеть так:

Спр1 = СоздатьОбъект("Справочник.Товары");

Спр1.ВыбратьЭлементы();

Пока Спр1.ПолучитьЭлемент() = 1 Цикл

Обработать(Спр1.ТекущийЭлемент());

КонецЦикла;

Функция ТекущийЭлемент() в качестве значения возвращает точное значение элемента справочника, на который спозиционирована выборка. А если функция ТекущийЭлемент() применяется к переменной, значением которой является уже точное значение элемента справочника, эта функция вернет само это значение. Так что равноценным вариантом решения показанной в примере проблемы было бы, если для выполнения действий процедура Обработать() использовала бы не сам переданный параметр, а значение некоторой переменной (например, Спр), значение которое получила бы из переданного параметра (Парам) следующим способом:

Спр = Парам.ТекущийЭлемент();

Однако следует иметь в виду, что при вызове метода ТекущийЭлемент() у переменной, значение которой получено из реквизитов форм и атрибутов объектов информационной базы, а не путем вызова функции СоздатьОбъект() будет выполняться поиск элемента справочника в информационной базе, что повлечет к дополнительным затратам времени. Соответственно, если полученное конкретное значение предполагается просто присвоить (например, измерению регистра), а не обращаться к его атрибутам и методам, и известно, что это именно конкретное значение, а не выборка, использовать метод ТекущийЭлемент() не следует. Поэтому можно рекомендовать создавать процедуры ориентированные на получение в качестве параметра только конкретного значения, а при вызове процедуры с передачей в качестве фактического параметра выборки вызывать метод ТекущийЭлемент().

Обработать(Спр1.ТекущийЭлемент());

Аналогичные соображения справедливы для объектов типа документ и счет бухгалтерского учета. Для документов аналогом функции ТекущийЭлемент() является функция ТекущийДокумент(), а для счетов бухгалтерского учета - ТекущийСчет().

**Выбор разрядности для данных числовых типов**

Выбор разрядности для данных числовых типов При описании формата числового типа данных первая цифра означает полное число знакомест в представлении числа (с учетом позиции, занимаемой десятичной точкой, но без учета позиции под знак числа), вторая цифра означает число знакомест в дробной части числа. Поэтому, если мы задали формат числа (8.2), то это означает, что число будет представляться с пятью знаками до десятичной точки и двумя знаками после (плюс один символ на десятичную точку).

При установке формата числового типа данных, разрядность должна обеспечивать возможность хранения чисел необходимой величины. Например, если у нас реквизит "Цена" имеет формат (8.2), то при записи в него значения 10000 \* 26.85 мы получим переполнение числа (99999.99).

Аналогичные соображения следует учитывать при выборе формата таких данных как ресурсы регистров оперативного учета, реквизиты многострочных частей документов, для которых установлен флаг "Итог по колонке", суммы проводок и операций, и т. п.. Для указанных величин необходимо помнить, что поля, в которых хранятся значения перечисленных величин и поля, в которых размещаются итоговые значения (суммы) имеют одинаковый формат. Поэтому, для такого рода числовых величин в формате должно быть учтено не только максимальное значение отдельной величины, но и максимально возможные значения сумм. Так, например, при выборе разрядности ресурсов регистров оперативного учета следует устанавливать разрядность, которая сможет вместить максимально возможный оборот и остаток по данному ресурсу.

**Перебор объектов и групповое изменение данных**

Весьма часто при разработке и внедрении конфигураций возникает необходимость в разработке алгоритма, который выполняет обход объектов информационной базы и вносит в них некоторые изменения. При реализации такого алгоритма в определенных случаях возникают ситуации, когда после изменения объекта не выполняется продолжение выборки следующих объектов. Это объясняется тем, что последовательная выборка и изменение объектов выполняются непосредственно в информационной базе. В основном проблемы возникают в тех случаях, когда при записи изменяются именно те данные, по которым собственно и происходит отбор объектов. В результате текущий объект не должен больше относиться к выборке и само продолжение выборки не выполняется системой. Например, в процессе перебора подчиненных документов, у найденного документа изменяется реквизит, в котором собственно записано значение того документа, для которого выбираются подчиненные. Ситуация осложняется тем, что при использовании методов языка разработчик конфигурации не всегда может точно определить каким способом система будет выполнять перебор объектов.

Прежде всего, можно рекомендовать не выполнять изменение данных тем же объектом (переменной) которой выполняется выбора, а создать для этой цели отдельную переменную и позиционировать ее на выбранный объект информационной базы.

Чтобы однозначно избежать трудностей при разработке подобных алгоритмов имеет смысл сначала выполнить обход объектов по некоторому критерию и занести их в список значений, а затем позиционироваться на каждый объект, содержащийся в списке и выполнять необходимые изменения.

Следует заметить, что при выполнении выборки объектов информационной базы с помощью объекта "Запрос" подобных проблем не возникает, так как при обходе результатов запроса используется временный массив построенный в ходе выполнения запроса.

**Ошибки базы данных и их обработка**

Ошибки базы данных и их обработка. При работе системы 1С:Предприятие могут возникать ошибки базы данных. Эти ошибки могут иметь место как при работе с базой данных, представленной в формате DBF/CDX , так и с базой данных, хранимой в среде MS SQL Server . И, хотя характер ошибок в обоих приведенных случаях может различаться, процедура их обработки стандартизована в системе 1С:Предприятие. По последствиям ошибки базы данных разделяются на две группы

невосстановимые ошибки

восстановимые ошибки

Невосстановимые ошибки базы данных

Невосстановимые ошибки базы данных являются следствием значительных нарушений в работе системы 1С:Предприятие: физического разрушения базы данных, нарушения ее структуры, серьезных рассогласований во взаимодействии программы с базой данных и т. п.. В таких случаях восстановить нормальное функционирование системы 1С:Предприятие невозможно и единственно правильной реакцией является завершение работы системы. Видимым проявлением таких ошибок в случае работы с базой данных в формате DBF/CDX является появление диалогового окна, содержащего текст, начинающийся со слов "Н евосстановимая ошибка Базы Данных”, в след за которыми приводится описание ошибки. В случае работы с базой данных, размещенной в среде MS SQL Server сначала появляется окно, в котором содержится описание ошибки SQL , а затем окно, содержащее текст "Н евосстановимая ошибка базы данных!". П осле этого в обоих случаях происходит завершение работы системы. Невосстановимые ошибки базы данных не могут быть перехвачены и обработаны из встроенного языка с помощью конструкций Попытка … Исключение … КонецПопытки .

Восстановимые ошибки базы данных

Восстановимые ошибки базы данных в известном смысле ошибками не являются, а представляют собой штатные ситуации, которые так или иначе могут возникать в процессе нормальной работы системы 1С:Предприятие. К восстановимым ошибкам относятся ошибки блокировки отдельных объектов (записей) базы данных, ошибки захвата таблиц при выполнении транзакций и т. п.. Ошибками они являются в том смысле, что при этом не могут быть успешно завершены выполняемые над базой данных операции (транзакции). Но при этом не происходит каких-то фатальных нарушений в работе системы 1С:Предприятие и имеется возможность восстановления нормального функционирования. Реакция системы на восстановимые ошибки базы данных зависит от того, в какой ситуации происходит ошибка: при выполнении интерактивного действия или при выполнении модуля встроенного языка.

При выполнении интерактивной процедуры (например, при записи документа из формы при нажатии кнопки), в случае возникновения восстановимой ошибки базы данных происходит откат транзакции базы данных, а сама система 1С:Предприятие возвращается к состоянию, в котором она находилась до начала выполнения интерактивной процедуры. После этого на экран выводится диалоговое окно, содержащее вопрос: "П ри выполнении транзакции произошла ошибка! <О писание ошибки> П овторить попытку выполнить транзакцию?". В случае положительного ответа на вопрос производится повторная попытка выполнить операцию. Если повторная попытка выполнить операцию снова завершится восстановимой ошибкой, то снова будет выдан запрос на повторение операции и так далее. В случае отрицательного ответа повторная попытка выполняться не будет. В модулях встроенного языка восстановимые ошибки базы данных могут быть перехвачены и обработаны с помощью конструкции встроенного языкаПопытка … Исключение … КонецПопытки так же, как и другие ошибки встроенного языка. В случае, если обработки ошибки в модуле не предусмотрено, то будет произведен откат выполняемой в данный момент транзакции, в окно сообщений 1С:Предприятия будет выведено сообщение о произошедшей ошибке, а выполнение модуля будет прервано, как и в случае возникновения в процессе выполнения модуля любой другой ошибки. В случае работы с базой данных, размещенной в среде MS SQL Server , в окно сообщений будет выдана ошибка SQL. После возникновения восстановимой ошибки алгоритм, вызвавший ошибку, может быть вызван пользователем повторно. Если используется конструкция встроенного языка Попытка … Исключение … КонецПопытки, то повторная операции записи может быть вызвана модулем автоматически.

**Использование строковых реквизитов неограниченной длины**

При определении структуры справочников и документов для строковых реквизитов необходимо выбрать определенную длину или указать режим "Неограниченная длина" (кроме реквизитов табличной части). Вариант "Неограниченная длина" существенно отличается по внутреннему способу хранения.

При работе с базами данных в формате DBF/CDX реквизиты неограниченные длины хранятся в отдельной таблице блоками по 80 символов. Для пустой строки в этой таблице записи не хранятся. Разумеется, хотя само значение блока имеет длину 80 символов, дополнительные ресурсы расходуются на поддержание связи с этой таблицей. В базах данных в формате SQL для хранения реквизитов неограниченной длины используется соответствующий тип поля таблицы, который также хранит только заполненные строки. Соответственно, если в создаваемом реквизите значения будут задаваться далеко не для всех объектов, то использование реквизита неограниченной длины может существенно уменьшить количество используемого дискового пространства.

Обращение к реквизитам неограниченной длины несколько замедляет считывание объекта, если происходит обращение к этим реквизитам. Запись объекта замедляется несколько более существенно, но только если конкретное значение изменялось. Скорость записи, разумеется, зависит от количества записываемых блоков (в базах данных DBF/CDX), то есть от длины строки.

Следует учитывать, что строковые значения реквизитов неограниченной длины записываются и возвращаются без хвостовых пробелов, тогда как обычные строковые реквизиты всегда дополняются пробелами до длины соответствующей длине реквизита в метаданных.

**Особенности работы с типизированными и не типизированными колонками таблицы значений**

При создании колонки таблицы значений можно указать или не указать тип значений, которые будут храниться в данной колонке. Информация в типизированных и не типизированных колонках таблицы значений хранится по-разному. В колонке с неуказанным типом значения хранится значение и его контекст, тогда как в колонке с указанным типом хранится только его значение. У каждого способа есть свои плюсы и минусы. Хранение значения и контекста позволяет увеличить производительность при частых обращениях к значениям реквизитов объектов, которые хранятся в таблице значений. Но при этом расходуется много памяти на хранение каждого объекта. Хранение только значения позволяет более компактно хранить информацию. Минусом является то, что при каждом обращении к атрибуту или методу объекта, который хранится в такой колонке, объект будет создаваться, позиционироваться на соответствующую запись и уничтожаться после использования. Чтобы избежать таких накладных расходов, перед обращением к нескольким атрибутам или методам объекта, рекомендуется поместить его значение во временную переменную и использовать ее для доступа к атрибутам и методам объекта. Данный способ позволяет компактно хранить информацию и получить хорошую производительность при работе с большим объемом информации.

**Применение процедуры ПриНачалеРаботыСистемы() глобального модуля**

В глобальном модуле может быть определена процедура ПриНачалеРаботыСистемы(), которая вызывается на выполнение при запуске системы 1С:Предприятие. Данная процедура предназначена для того, чтобы в конфигурации можно было определить действия, которые должны быть выполнены при начале работы системы 1С:Предприятие. Однако операторы тела глобального модуля (операторы, размещаемые в конце модуля и не входящие в какую-нибудь процедуру) также выполняются при запуске системы 1С:Предприятие. При выполнении тела глобального модуля сам глобальный модуль еще не полностью инициализирован, а если точнее, то его экспортируемые переменные, процедуры и функции на данный момент еще не доступны извне самого глобального модуля. И если в теле глобального модуля, например, с помощью функции ОткрытьФорму() запустить какую-либо обработку, то данная обработка не сможет быть выполнена, если в ней содержатся обращения к переменным, процедурам или функциям глобального модуля.

Процедура ПриНачалеРаботыСистемы() вызывается после выполнения тела глобального модуля. При вызове данной процедуры глобальный модуль уже полностью инициализирован и доступен для обращения из других модулей. Поэтому в данной процедуре можно выполнять действия запускающие другие модули, из которых осуществляется обращение к объектам глобального модуля.

**Доступ во встроенном языке к метаданным**

В 1С:Предприятии 7.7 существует возможность обращаться к объектам метаданных средствами встроенного языка. Эта возможность используется, прежде всего, для создания универсальных алгоритмов, не зависящих от конфигурации и конкретных объектов метаданных.

Весь доступ к метаданным осуществляется через объект "Метаданные" и иерархию подчиненных ему объектов. Фактически объект "Метаданные" является корнем иерархической системы объектов, которые во многом похожи на дерево метаданных в конфигураторе.

В отличие от других средств встроенного языка доступ к метаданным не имеет подробного описания в документации. Это объясняется тем, что названия атрибутов и методов совпадают с терминами, выводимыми в текстовом представлении структуры метаданных (меню "Конфигурация – Описание структуры метаданных"). Соответственно для ознакомления с составом атрибутов и методов следует получить текстовое представление и определить в нем необходимые термины. Специфические атрибуты и методы (не отображаемые в текстовом представлении) отражены в документации и Синтакс-Помощнике.

Обращение к объектам метаданных выполняется с помощью методов вышестоящего объекта. Название метода совпадает с названием типа объекта метаданных. В качестве параметра может быть передан идентификатор искомого объекта или его номер. Если объект найден, то возвращается собственно объект метаданных. Если объект не найден то возвращается пустой объект метаданных. Для проверки того, возвращен пустой объект или нет, следует использовать метод "Выбран".

Например:

Если Метаданные.Документ("Счет").Выбран()=0 Тогда

Сообщить("В конфигурации нет документа Счет");

КонецЕсли

Если параметр не задан, то возвращается количество объектов метаданных данного типа.

Например:

Сообщить("В конфигурации "+Метаданные.Документ()+" документов");

Эта возможность используется для обхода всех объектов метаданных

определенного типа.

Например:

Для Инд=1 По Метаданные.Документ() Цикл

Сообщить("Документ "+Метаданные.Документ(Инд));

КонецЦикла;

Для доступа к объектам метаданных, которые подчинены полученным объектам, также соответственно используются методы, совпадающие с именами метаданных. Заметим, что реквизит документа также является объектом метаданных (имеет идентификатор, синоним и т.д.).

Например:

Если

Метаданные.Документ("Счет").РеквизитШапки("Заказчик").Выбран()=0 Тогда

Сообщить("В конфигурации нет реквизита Заказчик документа

Счет");

КонецЕсли;

Для доступа к свойствам объектов метаданных используются атрибуты с соответствующими именами.

Например:

Сообщить("Тип номера расходного ордера:

"+Метаданные.Документ("РасходныйОрдер").ТипНомера);

Таким образом, если ознакомиться с содержимым текстового представления метаданных, то можно определить названия методов и атрибутов для доступа ко всем объектам метаданных и их свойствам.

Некоторые свойства объектов метаданных являются ссылками на другие объекты. В этом случае возвращается объект метаданных, к которому можно обращаться также как и объектам, полученным с помощью методов объекта "Метаданные".

Например:

Сообщить("Длина кода справочника владельца справочника Договора:

"+Метаданные.Справочник("Договора").Владелец.ДлинаКода);

Для некоторых объектов их свойства могут содержать несколько ссылок на различные объекты метаданных. В этом случае к ним применяются методы "Количество" и "Получить".

Например:

Сообщить("Количество ссылок в графе Сумма журнала Общий :"

+Метаданные.Журнал("Общий").Графа("Сумма").Ссылки.Количество());

Заметим, что возвращаемые значения свойств объектов метаданных, соответствующие одному из нескольких возможных вариантов, возвращаются в строковом виде в соответствии с установленном для конфигурации языком (русским или английским). Возможные значения в английском варианте можно посмотреть, сформировав текстовое описание метаданных при установленном в конфигурации английском языке.

Следует учитывать, что механизм обращения к метаданным не связан непосредственно с данными и не позволяет получать значения, хранящиеся в информационной базе или значения агрегатных типов которые могут храниться в информационной базе. Он предназначен только для получения информации о структуре метаданных. То есть все объекты, получаемые через объект "Метаданные", имеют специальный тип "Метаданные". Например, используя доступ к метаданным нельзя получить значение типа "Перечисление". То есть объект метаданных может сообщить свойства перечисления, но не может выдать собственно значение типа "Перечисления" чтобы присвоить его, например, реквизиту документа или сравнить с существующим значением. Для получения собственно значения типа "Перечисление" необходимо воспользоваться самим объектом "Перечисление". Однако существует возможность искать значение перечисления по идентификатору, полученному из метаданных.

Например:

Для Инд=1 По Метаданные.Перечисление("ВидыПлатежа").Значение() Цикл

Сообщить(Перечисление.ВидыПлатежа.ЗначениеПоИдентификатору(

Метаданные.Перечисление("ВидыПлатежа").Значение(Инд).Идентификатор));

КонецЦикла;

Объекты, используемые для доступа к метаданным, могут запоминаться в переменных для сокращения в дальнейшем доступа к их свойствам и подчиненным объектам.

Например:

МДСчет= Метаданные.Документ("Счет");

Сообщить("Тип реквизита Заказчик документа Счет:

"+МДСчет.РеквизитШапки("Заказчик").Тип);

Заметим, что объекты, используемые для доступа к метаданным, не могут сравниваться между собой. Для того чтобы сравнить, например, два объекта метаданных одного и того же типа на идентичность можно сравнить их идентификаторы.

**Особенности работы с реквизитами неопределенного типа**

В 1С:Предприятии 7.7 существует возможность создавать в объектах метаданных реквизиты неопределенного типа. Например, эта возможность активно используется для хранения в документе значений субконто связанных с некоторым счетом, по которому должны формироваться проводки документом.

Особенность использования реквизитов неопределенного типа заключается в том, что их значения нельзя присваивать, не назначив предварительно тип. Тип назначается с помощью метода "НазначитьТип" того объекта, которому принадлежит реквизит. Таким образом, реквизиты неопределенного типа отличаются от обычных переменных модулей, тип которых может свободно изменяться при присвоении значения.

Замечание!

В модуле формы документа (аналогично и справочника) для назначения типа реквизиту документа следует вызывать метод "НазначитьТип" у самого документа (то есть не через точку, а непосредственно). Вызов метода "НазначитьТип" для элемента управления (через объект Форма) следует применять только к реквизитам диалога формы (не являющимися реквизитами документа).

Следует учитывать, что реквизиты, имеющие неопределенный тип, не могут использоваться для хранения строк длиннее 22 символов.

**Особенности написания тела модуля**

При разработке модулей следует учитывать, что тело модуля (часть модуля, которая располагается ниже всех процедур и функций) выполняется системой при инициализации модуля. Эта особенность определяет некоторые ограничения на использование тела модуля.

Фактически тело модуля служит только для инициализации переменных модуля объявленных в его верхней части.

Для модуля формы в теле модуля не могут выполняться некоторые методы, которые управляют элементами управления и свойствами формы. Все эти действия рекомендуется выполнять в процедуре "ПриОткрытии".

Для глобального модуля в теле модуля также могут не выполняться некоторые функции встроенного языка. Например, в теле модуля не допускается открытие форм.

Для модуля документа если документ проводится из формы документа, то тело модуля будет выполнено только один раз, хотя проведение может быть выполнено несколько раз без закрытия формы. При групповом проведении тело модуля не будет выполняться перед началом проведения каждого документа (групповое проведение документов выполняется системой при переносе точки актуальности вперед и в режиме "Проведение документов"). Таким образом, переменные, которые должны инициализироваться при каждом проведении следует инициализировать в процедуре "ОбработкаПроведения", а не в теле модуля.