Параметры

Данный раздел содержит описания параметров, переменных, выражений и функций.

Раздел ‘*Параметры*’ может находиться либо на верхнем уровне любой библиотеки, либо внутри объектных разделов. Если раздел ‘*Параметры*’ расположен на верхнем уровне библиотеки, то в этом разделе нельзя описывать параметры объектов. Этот раздел предназначен только для описания переменных, выражений и функций.

В целях облегчения администрирования и ориентирования в дереве проекта были составлены правила заполнения раздела ‘*Параметры*’ для различных библиотек.

Правила для описания параметров объектов

- Объектные разделы в разделе ‘*Системная область*’ содержат описания стандартных параметров объектов. Описания основных таблиц БД обязательно должны включать стандартные параметры.
- Описания остальных параметров объектов находится в декларативных библиотеках, таких как [КЛАССЫ, ТИПЫ, ПАРАМЕТРЫ, ПЛАНЫ, ОПИСАНИЯ СПИСКОВ].

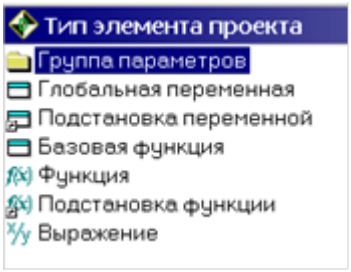
> Не рекомендуется бездумное создание новых параметров. Следует помнить, что единый список параметров значительно облегчает решение задачи совместимости разных проектов. Кроме того, у Oracle имеется ограничение на 1000 параметров в таблице.

Правила для описания переменных, выражений и функций

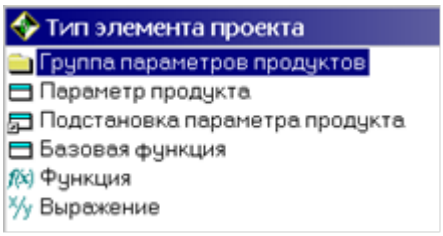
- Раздел ‘*Системная область*’ содержит описания контекстных и глобальных переменных. Также здесь находятся описания базовых функций, системных функций и выражений.
- Библиотека ‘*!Примитивы скриптов*’ содержит описания системных функций, реализованных на языке скриптов.
- Параметры из остальных системных библиотек созданы, прежде всего, для их использования в этих библиотеках.
- В библиотеку ‘!*Базовый набор*’ вынесены описания тех глобальных переменных, выражений и функций, которые имеют некое прикладное значение.
- Проектные базовые и проектные функциональные библиотеки содержат описания функций и выражений, являющихся общими для группы проектов.
- Функции и выражения из проектных библиотек предназначены для использования в проектах.

Атрибуты раздела ‘Параметры’

Внутри раздела ‘*Параметры*’ на верхнем уровне библиотеки можно создать элементы следующих типов:



Для раздела ‘*Параметры*’ из объектного раздела список возможных элементов меньше. Например внутри раздела ‘*Продукт*’ можно создать элементы следующих типов:



- **Группа параметров** – для создания папки, в которой могут быть сгруппированы несколько параметров, функций или выражений, объединенных по какому-либо признаку. Для группы указывается только наименование.
- **Глобальная переменная** – для описания новой глобальной переменной.
- **Подстановка переменной** – для изменения свойств глобальной или контекстной переменной.
- **Базовая функция** – для описания новой базовой функции.
- **Функция** – для описания новой функции.
- **Подстановка функции** – для изменения свойств функции.
- **Выражение** – для описания нового выражения.
- **Параметр <объекта>** – для описания нового параметра указанного объекта.
- **Подстановка параметра <объекта>** – для изменения свойств параметра объекта.

Переменные в проекте

Переменная – это хранилище некой информации. Переменные всегда имеют имена. В процессе работы программы информация в переменной (хранилище) может изменяться.

В проекте имеются следующие виды переменных:

- глобальные переменные;
- контекстные переменные;
- локальные переменные.

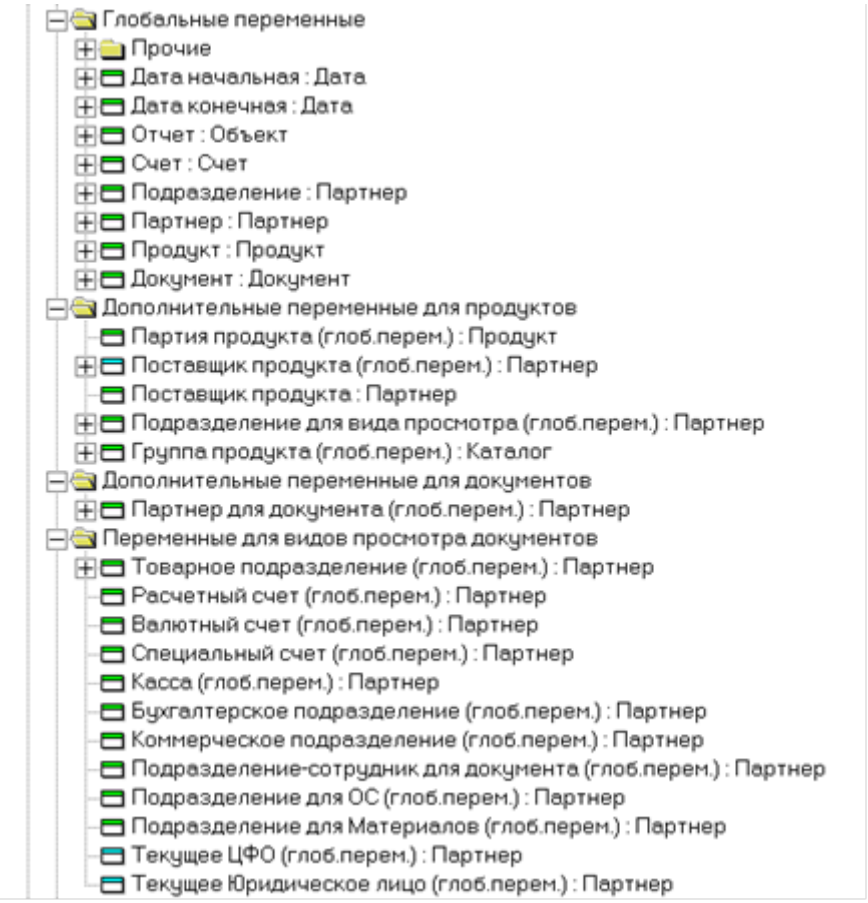
Все переменные в дереве проекта обозначены иконкой  .

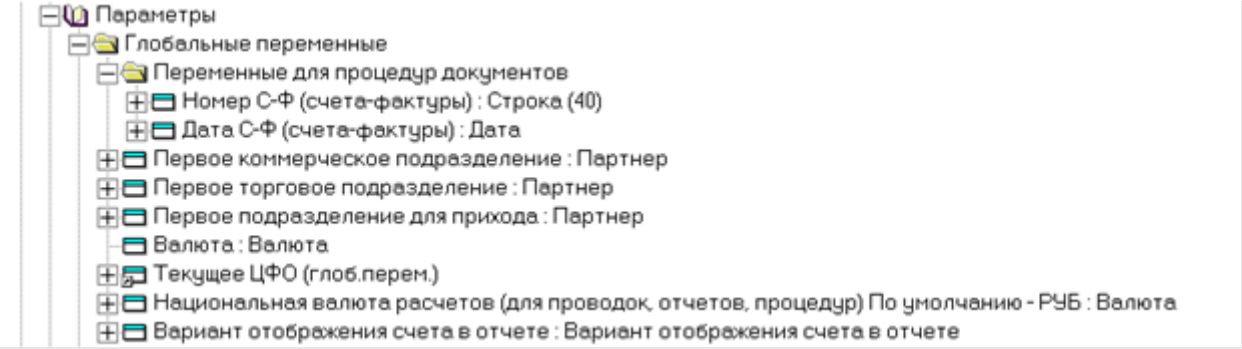
Цвет иконки зависит от статуса проектного элемента. Если проектировщик присвоил переменной статус ‘Экспортный (Публичный’), то отображается иконка зеленого цвета. Для непубличных переменных используется иконка синего цвета.

Глобальные переменные

В Домино имеется набор переменных, доступных в любом месте проекта. Именно по причине их общей доступности такие переменные названы глобальными. Глобальным переменным можно свободно присваивать значения и использовать их в выражениях, функциях и процедурах, расположенных в любых библиотеках и ‘Приложении’.

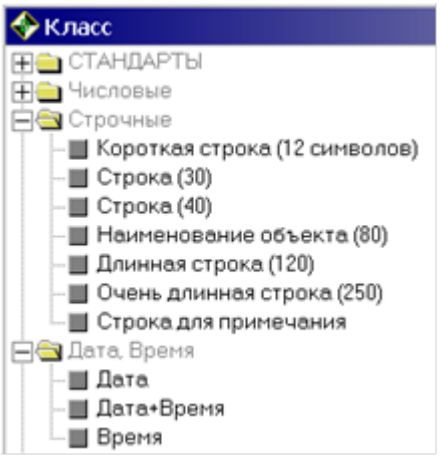
Обычно глобальные переменные задаются либо в базовом разделе ‘*Системная область*’, раздел ‘*Параметры*’, папка ‘*Глобальные переменные*’, либо в библиотеке ‘*!Базовый набор*’, раздел *‘Параметры*’, папка ‘*Глобальные переменные*’.





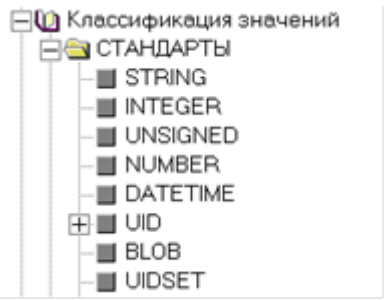
В форме глобальной переменной заполняются наименование переменной и класс значения.

Класс значения определяет тип данных, записываемых в переменную. Описания основных типов данных находится в базовом разделе ‘Системная область’ раздел ‘Классификация значений’.



Типы данных подразделяются на две группы. Первую группу составляют стандартные типы данных. Вторая группа состоит из производных типов, которые создаются на основе стандартных.

Рассмотрим подробнее стандартные типы данных.

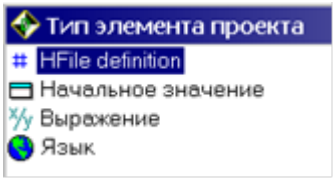


- **STRING (строка) -** это последовательность символов произвольной длины. При вычислении выражений нужный размер строка принимает автоматически.
- **INTEGER (целое) -** целое число, положительное, нуль или отрицательное. Может принимать значения от -2’147’483’648 до 2’147’483’647.
- **UNSIGNED (целое без знака) –** положительное целое число или нуль. Может принимать значения от 0 до 4’294’967’295.

- **NUMBER (число с дробной частью) -** десятичное число с фиксированной дробной частью. Максимальный размер 24 цифры, включая десятичные. Если в описании не указаны ни число разрядов, ни точность, то устанавливается так называемая ‘плавающая точность’. Плавающая точность означает, что точность числа может изменяться для того, чтобы сохранить как можно больше значащих цифр после десятичной точки.
- **DATETIME (дата, время) -** дата со временем с точностью до секунды.
- **UID -** уникальный идентификатор объекта базы данных или проектного элемента**.**
- **BLOB –** данные большого объема, хранящиеся в двоичном виде.
- **UIDSET** – список уникальных идентификаторов (UID) объектов.

Тип данных определяет множество значений и операции, которые могут быть применимы к этим значениям.

Для глобальной переменной могут быть заданы следующие атрибуты:



- **HFile definition** – используется программистами.
- **Начальное значение** - присваивается при старте Домино. Если начальное значение не указано, то переменная имеет значение NULL.

> При завершении работы Домино текущие значения глобальных переменных сохраняются в специальном файле. При последующем старте Домино глобальным переменным присваивают сохраненные значения. Значение из файла имеет приоритет над значением, заданным в данном атрибуте.

- **Выражение** – рассчитывается при старте Домино и результат присваивается глобальной переменной.

> Выражение рассчитывается после присваивания начального значения, указанного в предыдущем атрибуте, и после считывания значения, сохраненного в специальном файле. Т.е. если для глобальной переменной задан атрибут ‘*Выражение*’, то значение глобальной переменной **при старте** Домино **всегда** будет рассчитано как результат указанного выражения.

- **Язык** – применяется для поддержки национальных языков. Подробное описание назначения данного атрибута находится в главе ‘**Виды просмотра**’.

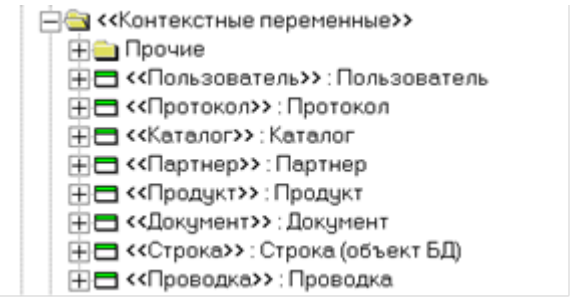
Подстановка глобальной переменной применяется для изменения значений указанных атрибутов.

Контекстные переменные

Замысел появления контекста в Домино заключался в том, чтобы с помощью контекста точнее понимать смысл действий программы в отношении обрабатываемых данных. Иными словами, проектировщик, анализируя контекст, понимает какие именно объекты обрабатываются в рассматриваемом месте программы. Для того чтобы контекстом было удобно пользоваться, контекст должен содержать как можно меньше частей. Этого можно добиться, если для каждого объекта завести свою переменную, в которую записывать UID текущей записи объекта.

Контекст – это набор значений специальных (**контекстных**) **переменных**, содержащих идентификаторы текущих записей основных объектов БД.

Контекстные переменные перечислены в библиотеке ‘Системная область’, раздел ‘Параметры’, папка ‘Контекстные переменные’.



Список содержит контекстные переменные по всем объектам.

> В папке ‘Прочие’ находятся дополнительные контекстные переменные. Дополнительные контекстные переменные были созданы до того, как в Домино появились локальные переменные и формальные параметры. В настоящее время имеются более удобные и понятные средства для реализации тех задач, в которых могут потребоваться эти переменные, поэтому **использование дополнительных контекстных переменных не рекомендуется**.

Форма и атрибуты контекстной переменной совпадают с формой и атрибутами глобальной переменной.

Контекстные переменные доступны в любом месте проекта, если только область видимости специально не ограничена проектировщиком. Контекстные переменным можно свободно присваивать значения и использовать их в выражениях, процедурах и функциях.

Контекстным переменным отводится следующая роль:

- Значения контекстных переменных (контекст) содержат идентификаторы текущих объектов БД.

- Посредством контекстных переменных осуществляется доступ к параметрам текущих записей объектов. Например, конструкция <<Продукт>>.Имя определяет наименование продукта из текущей записи продукта.
- В видах просмотра в контекстную переменную отображаемого объекта заносится идентификатор той строки, на которой находится курсор. Например, в виде просмотра партнеров заполняется переменная <<Партнер>>.
- Некоторые функции и процедуры используют контекстные переменные в качестве неявных параметров. Такие процедуры называются контекстно-зависимыми.
- Контекст применяется при вычислении выражений. Для примера рассмотрим вычисление конструкции Продукт.Имя & Партнер.Имя. Если выражение вычисляется от текущей записи продукта, то для получения записи партнера используется контекстная переменная <<Партнер>>. Если данное выражение вычисляется для текущей записи партнера, то ссылка на запись продукта считывается из контекстной переменной <<Продукт>>. В обоих случаях идентификатор записи получен из контекстной переменной.

> Несмотря на то, что на уровне ядра Домино имеется полная поддержка контекстных переменных, использование контекста не является общим для всех компонент программы. Контекстные переменные поддерживаются во всех видах просмотра. Разработчики других компонент (форм, методов акцепта, процедур и т.д.) посчитали излишним обеспечение контекста в своих компонентах.

> Не рекомендуется создавать контекстные переменные без острой необходимости, поскольку обслуживание контекста требует значительных ресурсов.

Подстановка контекстной переменной применяется для изменения значений атрибутов.

Локальные переменные

Локальная переменная – это переменная, действующая только внутри процедуры, функции или выражения.

При описании процедуры (функции, выражения) можно указать необходимые локальные переменные в разделах ‘Локальные переменные’ или ‘Формальные параметры’. Здесь же задаются начальные значения переменных.

При запуске процедуры (функции, выражения) ядро программы создает указанные локальные переменные и присваивает им начальные значения. По завершении процедуры все локальные переменные удаляются.

Если локальные переменные расположены в разделе ‘Формальные параметры’, то это означает, что процедура имеет соответствующие параметры для вызова.

Значения формальных параметров, указанные при вызове процедуры, имеют приоритет над

начальными значениями в описании. Т.е. при старте процедуры в локальную переменную (являющуюся формальным параметром) будет записано значение при вызове, а если таковое отсутствует, то будет записано начальное значение из описания.

Описание локальной переменной похоже на описание глобальной и контекстной переменных.

Параметры объектов

В Домино имеются следующие основные таблицы БД для хранения объектов:

- **Права** – для хранения назначенных пользователю ролей;
- **Нумератор** – для нумерации объектов;
- **Протокол** – для хранения сообщений;
- **Пользователь** – для хранения списка пользователей;
- **Каталог** – для хранения каталогов (товарные группы, регионы, виды продукции и другие);
- **Продукт** – для хранения учетных единиц;
- **Партнер** – для хранения торговых партнеров и подразделений;
- **Документ** – для хранения документов всех видов;
- **Строка** – для хранения строк документов;
- **Проводка** – для хранения проводок всех видов;
- **Сальдо** – для хранения сальдо;
- **Контрольная точка** – для хранения списка установленных контрольных точек;
- **Сальдо по контрольной точке** – для хранения сальдо по контрольным точкам.

> Таблица ‘Обороты’ реально не существует. Она является виртуальной сущностью (view), которая появляется только в процессе расчета оборотов. Эта таблица имеет описание в проекте только для того, чтобы проектировщики могли использовать параметры таблицы.

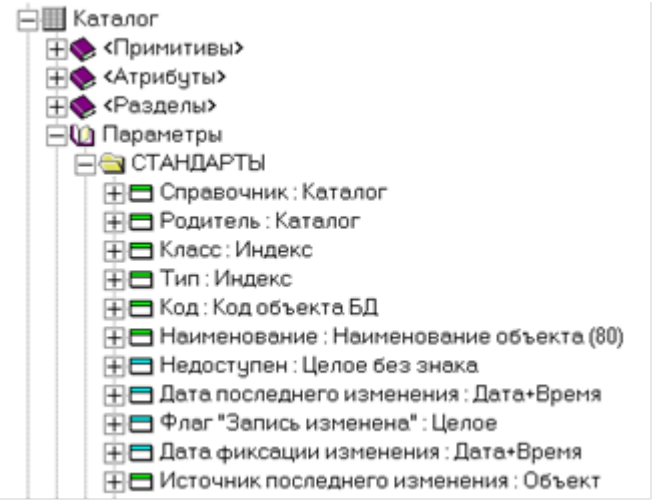
> Таблицы ‘*Сальдо*’, ‘*Контрольная точка*’ и ‘*Сальдо по контрольной точке*’ доступны только для чтения. Запись данных в эти таблицы осуществляется специальными средствами.

Каждая таблица содержит строки (записи) и столбцы (поля, параметры). Одному объекту соответствует одна запись таблицы. Характеристики объекта записываются в параметры таблицы.

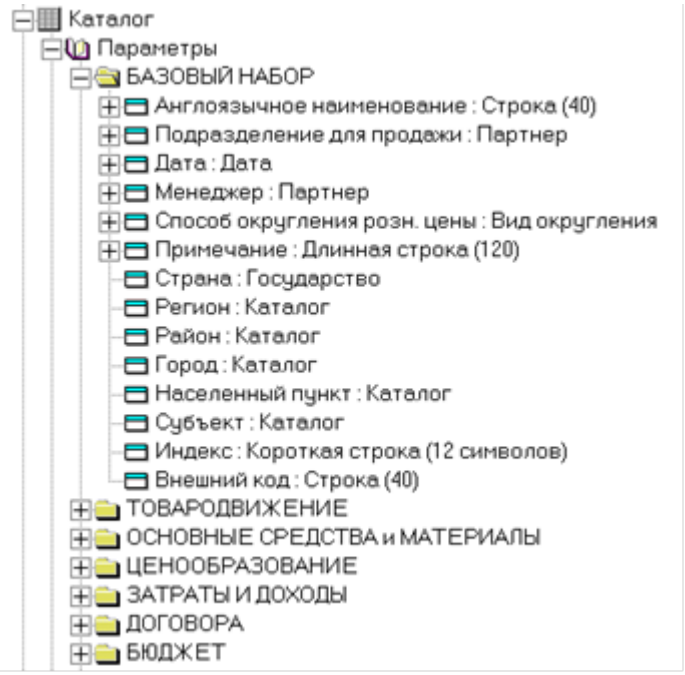
Все параметры необходимо зарегистрировать в проекте. Часть параметров являются обязательными. Такие параметры называются стандартными, они перечислены в базовом разделе ‘*Системная область*’ в объектных разделах.

Например, список стандартных параметров таблицы ‘*Каталог*’ находится внутри раздела ‘*Каталог*

’ в разделе ‘*Параметры*’.



Остальные параметры находятся в декларативных библиотеках. Например, в библиотеке [КЛАССЫ, ТИПЫ, ПАРАМЕТРЫ, ПЛАНЫ, ОПИСАНИЯ СПИСКОВ].



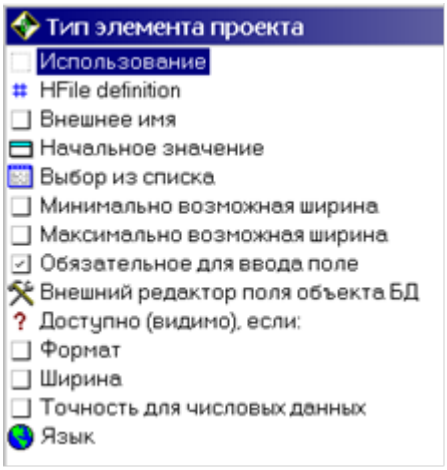
> Разрешено добавлять параметры только для таблиц *‘Пользователь’, ‘Протокол’, ‘Каталог’, ‘Партнер’, ‘Продукт’, ‘Строка’, ‘Документ’.*

Параметры в дереве проекта обозначены иконкой  .

В форме параметра заполняются наименование параметра и класс значения.

Класс значения определяет тип данных, записываемых в параметр. Подробнее о типах данных написано выше (текст о глобальных переменных).

При описании параметра можно использовать следующие атрибуты:



- **Использование** – условие доступа проектировщиков к параметру.
- **HFile definition** – используется программистами.
- **Внешнее имя** – имя столбца в БД. Если внешнее имя не указано, то имя столбца рассчитывается автоматически (F<индекс проектного элемента>).
- **Начальное значение** – присваивается при создании новой записи.
- **Выбор из списка** – для ссылки на вид просмотра, содержащий список возможных значений поля. Применяется в видах просмотра и формах. Если для поля вида просмотра или формы не указан аналогичный атрибут, то используется данное значение атрибута.
- **Минимально возможная ширина** – минимальная ширина колонки. Применяется в видах просмотра. Если для поля вида просмотра не указан аналогичный атрибут, то используется данное значение атрибута.
- **Максимально возможная ширина** – максимальная ширина колонки. Применяется в видах просмотра. Если для поля вида просмотра не указан аналогичный атрибут, то используется данное значение атрибута.
- **Обязательное для ввода поле** – при установленном признаке значение данного поля должно быть обязательно введено. Применяется в формах. Если для поля формы не указан аналогичный атрибут, то используется данное значение атрибута.
- **Внешний редактор поля объекта БД** – задает специальную функцию для ввода значения поля. Применяется в формах. Если для поля формы не указан аналогичный атрибут, то используется данное значение атрибута.
- **Доступно (видимо), если** – задает условие отображения параметра. Применяется в видах просмотра и формах. Если для поля вида просмотра или формы не указан аналогичный атрибут, то используется данное значение атрибута.
- **Формат** – для задания формата отображения содержимого параметра. Применяется в видах просмотра и формах. Если для поля вида просмотра или формы не указан аналогичный атрибут, то используется данное значение атрибута.
- **Ширина** – для задания ширины строкового поля или числа знаков числового поля. Данный атрибут изменяет
- **Точность для числовых данных** – задает число знаков после запятой для числовых полей. Применяется в формах. Если для поля формы не указан аналогичный атрибут, то используется данное значение атрибута.

- **Язык** – применяется для поддержки национальных языков. Применяется в видах просмотра и формах. Если для поля вида просмотра или формы не указан аналогичный атрибут, то используется данное значение атрибута.

> Описание атрибутов, типичных для формы и вида просмотра, вынесено на уровень параметра объекта для того, чтобы повторно не указывать одинаковые значения атрибутов.

Подстановка параметра применяется для изменения значений перечисленных атрибутов параметра.

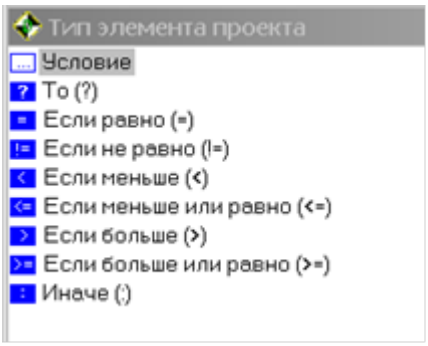
Выражения

Выражение состоит из одного или нескольких операндов и знаков операций. Операндами выражения могут быть константы, локальные, глобальные и контекстные переменные, другие выражения, а также - функции. Операции выполняются над операндами в порядке их следования. В Домино отсутствует понятие старшинства операций, и если требуется изменить порядок выполнения операций, то следует использовать вложенные выражения.

Выражения в дереве проекта обозначены иконкой  .

При описании выражения можно использовать следующие атрибуты:

- **Результат** – константа или ссылка на контекстную, локальную или глобальную переменные, другое выражение, функцию, параметр объекта базы данных. Если данный атрибут возвращает объект, то с помощью проектного элемента ‘Уточняющий параметр’ получается значение нужного параметра этого объекта. При необходимости можно пройтись по цепочке связанных объектов. Подробнее от уточняющем параметре написано в главе ‘Виды просмотра’.
- **Ноль (0.0), если NULL** - Возвращает значение контекстной, локальной или глобальной переменных, результат вычисления другого выражения, функции, содержимое параметра объекта базы данных. Если полученный результат имеет значение NULL, то возвращает ноль (0.0).
- **NULL** – Константа NULL.
- **Выражение** - Служит для изменения порядка вычисления операторов. Применение данной операции аналогично скобкам.
- **Оператор Выбора** - Позволяет указать, какую нескольких ветвей выражения вычислить в зависимости от значения указанного условия.

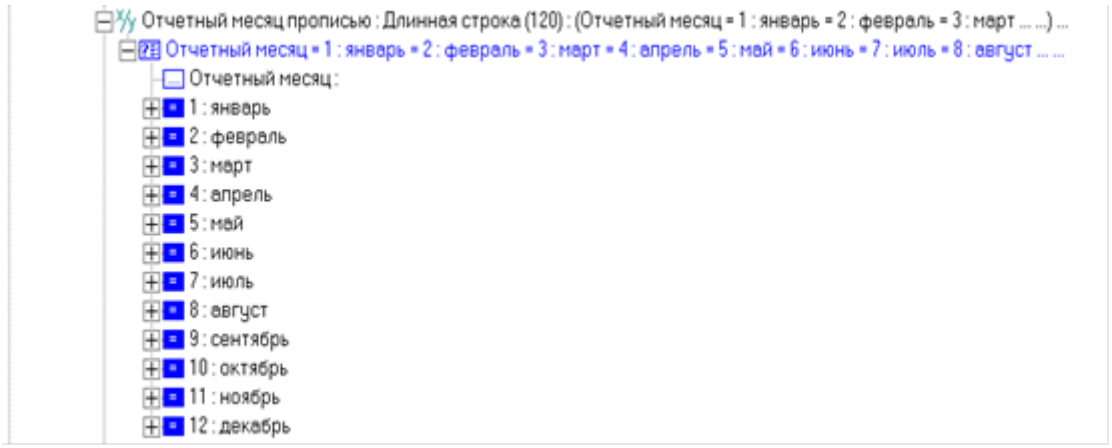


- **Условие** - это переменная (выражение или функция), значение которой будет являться ключом для поиска подходящего варианта.
- **То (?)** – Если условие имеет истинное значение (TRUE), то вычисляется последовательность операндов этого раздела и выход из оператора выбора.
- **Если равно (=) , Если не равно (!=), Если меньше (<), Если меньше или равно (<=), Если больше (>), Если больше или равно (>=)** - Для каждой альтернативы указываются значения-ключи. Эти значения-ключи сравниваются со значением условия по указанному правилу. Сравнение производится до первого выполнения правила. Как только между условием и значением-ключом выполняется указанное правило, то вычисляется последовательность операндов, заданная в разделе соответствующего правила. Затем выполняется выход из оператора выбора.
- **Иначе (:)** – Последовательность операндов данного раздела вычисляется в том случае, если ни одно правило сравнения условия и значений-ключей не было выполнено.

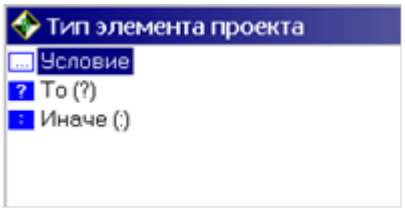
Если отсутствует условие ‘Иначе’ и ни одно из правил сравнения не было выполнено, то результат равен NULL.

Не рекомендуется использовать оператор выбора вместо условного оператора IF. Данная конструкция удобна для сравнения условия с набором явно заданных констант, классификаторов и кодификаторов.

Пример использования:



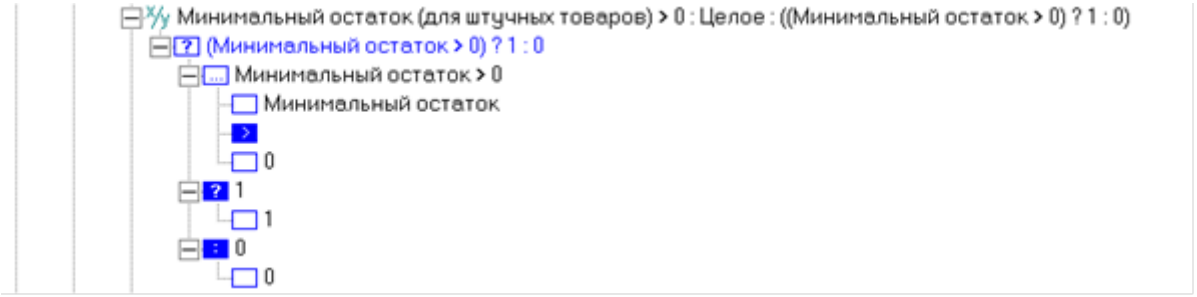
- **Условный оператор (IF)** - Позволяет указать, какую из двух ветвей выражения вычислить в зависимости от значения (истинное или ложное) указанного условия.



  - **Условие** - это переменная (выражение или функция), имеющая либо истинное, либо ложное значение.
  - **То (?)** - последовательность операндов данного раздела вычисляется если условие имеет истинное значение.
  - **Иначе (:)** - последовательность операндов данного раздела вычисляется если условие имеет ложное значение.

  Обязательное наличие сразу обоих разделов **То** и **Иначе** не требуется.

  Пример использования:



- **Конкатенировать строки (&) -** результатом является строка, полученная добавлением второй строки к первой.

Если между строчными операндами не указать операцию, то программа выполнит такую запись как конкатенацию строк.

- **Сложить (+) -** Результатом является сумма чисел.
- **Вычесть (-) -** Результатом является разность чисел.

Для переменных типа ДАТА перечисленные операции имеют особый смысл:

- **Дата (+) число** – результатом является дата, для которой время увеличено на число секунд.
- **Дата (-) число** - результатом является дата, для которой время уменьшено на число секунд.
- **Делить (/)** - Второй операнд воспринимается как делитель, а первый - как делимое. Результатом является частное от деления чисел.

- **Умножить (*)** – Результатом является произведение чисел.

Для переменных типа UIDSET (список уникальных идентификаторов) и UID перечисленные операции имеют следующий смысл:

  - **Сложение (+)** – Результатом является объединение двух списков.
  - **Вычитание (-)** – Результатом является список, содержащий элементы, которые имеются в первом операнде, и не имеются во втором операнде.
  - **Деление (/)** – Результатом является список, содержащий элементы, которые либо имеются в первом операнде, но не имеются во втором операнде, либо имеются во втором операнде, но не имеются в первом операнде. Иными словами, результатом является объединение двух вычитаний, в которых операнды меняются местами.
  - **Умножение (*) -** Результатом является пересечение двух списков.
- **Прибавить процент (+%)** – К первому операнду добавляется процент от первого операнда. Количество процентов указано во втором операнде.
- **Вычесть процент (-%)** - Из первого операнда вычитается процент от первого операнда. Количество процентов указано во втором операнде.
- **Обратный процент (/%) –** Результатом является частное от деления первого операнда на процент от первого операнда. Количество процентов указано во втором операнде.
- **Процент (*%) -** Результатом является процент от первого операнда. Количество процентов указано во втором операнде.
- **Возвести в степень (**)** – Возвести операнд в указанную степень.
- **Корень квадратный (SQRT)** – Вычислить квадратный корень от операнда.

В дальнейшем тексте будут использованы понятия истинного и ложного значений. Уточним термины.

> **Истинное и ложное значения операндов**
>
> Для логических операций имеются понятия истинного и ложного значений. Коротко истинное значение называют TRUE, а ложное – FALSE.
>
> Считается, что числовой операнд имеет **истинное значение**, если число не равно ни 0, ни NULL. Числовой операнд имеет **ложное значение** в противном случае (т.е. равен либо 0, либо NULL).

> Операнд типа Строка имеет **истинное значение**, если строка не пустая. Операнд типа Строка имеет **ложное значение** в противном случае. Хвостовые пробелы при сравнении строк игнорируются.
>
> Операнд типа UID объекта имеет **истинное значение**, если UID не равен NULL. Операнд типа UID объекта имеет **ложное значение**, если UID равен NULL.

- **ИЛИ (||)** - логическое ‘ИЛИ’. Если любой из операндов имеет истинное значение, то результат тоже будет иметь истинное значение. В противном случае результат будет иметь ложное значение.
- **И (&&)** - логическое ‘И’. Если оба операнда имеют истинное значение, то результат тоже будет иметь истинное значение. В противном случае результат будет иметь ложное значение.
- **НЕ (!)** - логическое отрицание. Если операнд имеет истинное значение, то результат будет иметь ложное значение. Если операнд имеет ложное значение, то результат будет иметь истинное значение.
- **Равно (==)** - Результат будет иметь истинное значение, если значения операндов равны. Результат будет иметь ложное значение в противном случае.
- **Не равно (<>)** - Результат будет иметь истинное значение, если значения операндов не равны. Результат будет иметь ложное значение в противном случае.
- **Больше (>)** - Результат будет иметь истинное значение, если значение первого операнда больше значения второго операнда. Результат будет иметь ложное значение в противном случае.
- **Больше или равно (>=)** - Результат будет иметь истинное значение, если значение первого операнда больше или равно значению второго операнда. Результат будет иметь ложное значение в противном случае.
- **Меньше (<)** – Результат будет иметь истинное значение, если значение первого операнда меньше значения второго операнда. Результат будет иметь ложное значение в противном случае.
- **Меньше или равно (<=)** - Результат будет иметь истинное значение, если значение первого операнда меньше или равно значению второго операнда. Результат будет иметь ложное значение в противном случае.
- **Допускает наследование атрибутов** – при установленном признаке в описании наследника можно будет переопределить атрибуты.
- **Использование** – условие доступа проектировщиков к выражению.
- **Округление** – вариант округления результата.
- **Формальные параметры** – список формальных параметров для вызова выражения
- **Значения формальных параметров при вызове** – список значений формальных параметров. Раздел заполняется при вызове выражения.
- **Локальные переменные** – список локальных переменных.
- **Вывести результат вычисления на экран** – применяется для отладки выражений. При установленном признаке результат вычислений высвечивается на экране в отдельном окне.

- **Указатель на элемент** – применяется для указания специальных атрибутов выражения.

Функции

Функция – это группа операторов, исполняемая как единое целое и предназначенная для выполнения некоторых действий и/или получения значения. Любую функцию можно вызвать как процедуру, проигнорировав возвращаемое ею значение.

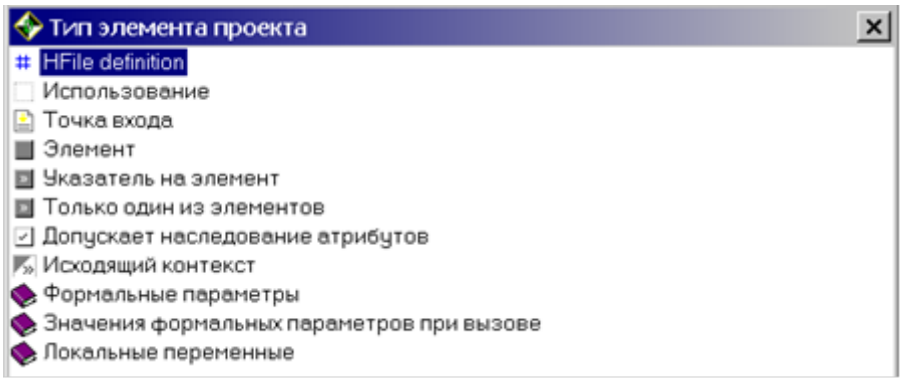
Результат, возвращаемый функцией, можно использовать в выражениях, при заполнении параметров процедур и функций, в условных и итерационных операциях.

Настроенные в проекте функции могут ссылаться на один из типов данных. При этом возвращаемое значение будет либо данного типа. Если у функции нет ссылки на тип, то тип возвращаемого значения определяется внутри функции.

Функции различаются по способу создания. Если функция разработана программистом на языке Visual C++, то код функции записан в динамической библиотеке. В проекте находится только декларация функции, причем атрибут ‘Точка входа’ содержит ссылку на динамическую библиотеку.

В другом случае функцию разрабатывает проектировщик на языке скриптов. Функции на языке скриптов целиком находятся в разделе проекта ‘Параметры’.

Для функций поддерживается механизм наследования. Функции самого верхнего уровня являются базовыми. У базовых функций имеются следующие атрибуты:

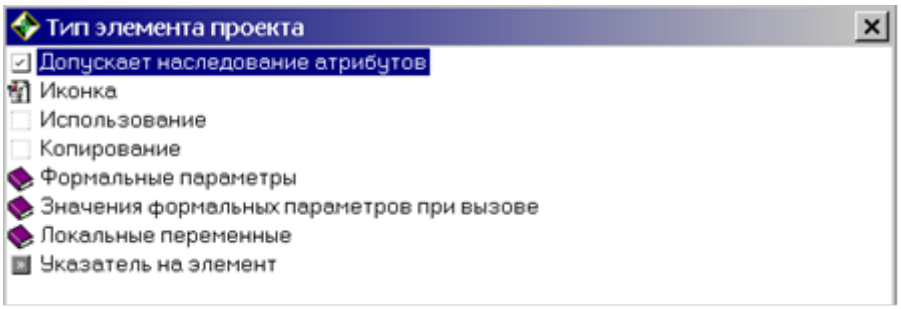


- **HFile definition** – используется программистами.
- **Использование** – условие доступа проектировщиков к функции.
- **Точка входа** – место реализации функции.
- **Элемент** – применяется для указания специальных атрибутов функции.
- **Указатель на элемент** – применяется для указания специальных атрибутов функции.
- **Только один из элементов** – не используется.
- **Допускает наследование атрибутов** – при установленном признаке в описании наследника можно будет переопределить атрибуты.
- **Формальные параметры** – список формальных параметров для вызова функции.
- **Значения формальных параметров при вызове** – список значений формальных

параметров. Раздел заполняется при вызове функции.

- **Локальные переменные** – список локальных переменных.

Для функций, не являющихся базовыми, можно указать следующие атрибуты:



Функции на языке скриптов имеют множество дополнительных атрибутов. Подробнее о таких функциях написано в книге '**Язык скриптов**'.

Функция может иметь параметры, которые требуется заполнить при ее вызове. Параметры могут быть обязательные, необязательные и неявные.

При вызове функции для обязательных параметров всегда должны быть указаны значения.

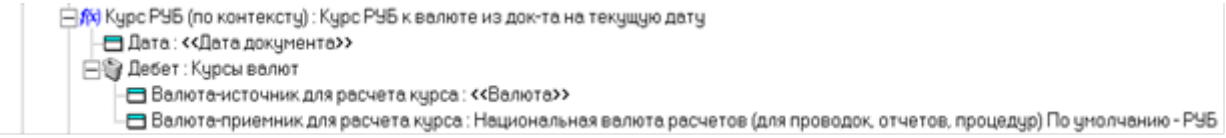
Для необязательных параметров значения можно не указывать. Обычно автор функции задает начальные значения таких параметров. Эти начальные значения и будут применены, если при вызове функции не указать иные значения.

Теперь разберем понятие неявных параметров функции. Параметр является неявным в том случае, если заполнение этого параметра происходит вне вызова функции.

> Наличие неявных параметров у функции значительно снижает понятность кода, поскольку проектировщику приходится помнить (или проверять по примерам) какие параметры имеются у той или иной встреченной им функции. С другой стороны, если при вызове функции не требуется указывать параметры, то код сократится. Что само по себе не так уж и плохо, поскольку длинный текст воспринимается хуже. Особенно, когда одна функция вызывается несколько раз с одинаковыми параметрами. В попытке найти разумную грань или компромисс и были созданы функции с неявными параметрами.

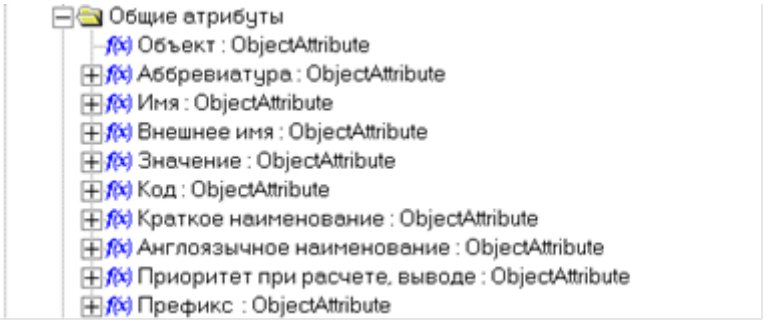
В проекте имеется множество функций, требующих заполнения соответствующих глобальных и контекстных переменных.

Функция называется **контекстно-зависимой** если функция имеет неявные параметры, и эти параметры являются контекстными переменными.



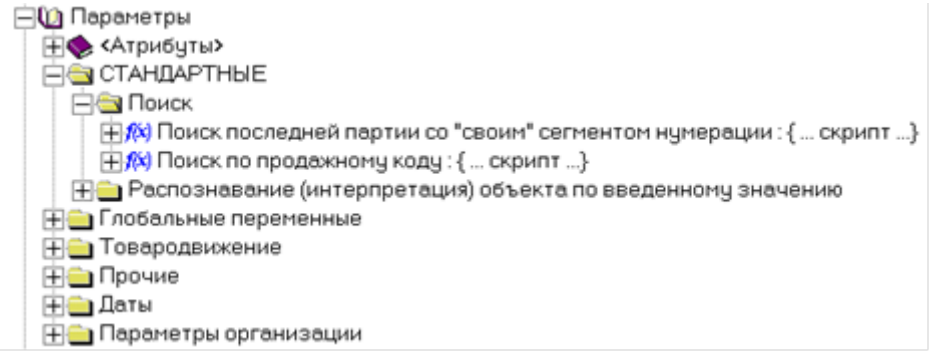
Функция 'Курс РУБ (по контексту)' имеет три неявных параметра: 'Дата документа', 'Валюта', 'Национальная валюта расчетов'. Два первых параметра являются контекстными переменными, а третий – глобальная переменная.

Еще один вариант функций с неявными параметрами – это **функции, работающие от текущего объекта**. Обычно такие функции вызываются как уточняющий параметр. Объектом может быть как запись таблицы БД, так и проектный элемент.



Создание группы параметров, объединенных в одной папке

Группа параметров создается для объединения в одной папке (под одним названием) нескольких параметров, переменных, функций, выражений. Это делается для того, чтобы было легче ориентироваться в многообразии элементов одного вида в дереве проекта.



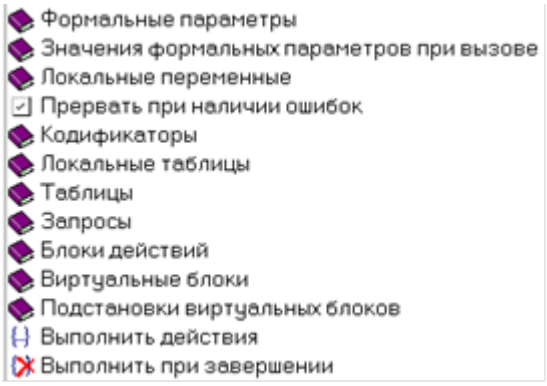
Группа параметров может содержать неограниченное число параметров, переменных, функций, выражений, Групп параметров и Подстановок переменных и функций.

Подстановка функции

Подстановка применяется только для функций на языке скриптов.

В подстановке можно добавить значения в описательные атрибуты функции и полностью изменить те атрибуты, которые задают алгоритм выполнения функции.

Рассмотрим основные атрибуты функции на языке скриптов.



Большинство атрибутов являются описательными. Если атрибут описан в подстановке, то значение из подстановки добавляется к значению соответствующего атрибута из описания функции.

Атрибуты ‘*Подстановки виртуальных блоков*’, ‘*Выполнить действия*’, ‘*Выполнить при завершении*’ задают алгоритм действий. Если атрибут описан в подстановке, то значение из подстановки заменяет значение соответствующего атрибута из описания функции.

Значение признака ‘*Прервать при наличии ошибок*’ в подстановке также заменяет значение этого признака в описании функции.

Версия #4

Демонов Сергей создал Wed, Mar 23, 2022 4:34 PM

Демонов Сергей обновил Thu, Mar 24, 2022 5:48 PM