Операции

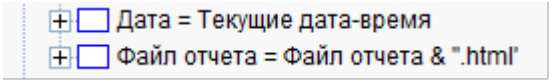
Операции предназначены для того, чтобы сообщить компьютеру, какое именно действие требуется выполнить. Операции в языках программирования принято делить на следующие виды:

- Операция присваивания – используется при записи значения в переменную;
- Операции, используемые в выражениях – для расчета результата выражения;
- Условные операции – содержат альтернативные ветви вычислений;
- Итеративные (повторяющиеся) операции – указывают компьютеру, что требуется повторить вычисления с указанного места;
- Вызов процедуры или функции – указывает компьютеру, что требуется выполнить набор операций, перечисленных в указанной процедуре (функции);
- Прочие операции – все те операции, которые невозможно отнести ни к одному из перечисленных видов.

В языке скриптов имеются операции всех перечисленных видов.

Оператор Присвоить

Операция присваивания применяется для записи значения в переменную. Значение может быть указано либо явно (с помощью константы), либо в виде выражения. Во втором случае сначала будет рассчитано выражение, затем полученный результат будет занесен в переменную. Также произойдет приведение типа результата выражения к типу переменной.



Разные варианты присваивания значений переменным (константы, результата выражения).

Операции, используемые в выражениях

Выражение состоит из одного или нескольких операндов и знаков операций. Операндами выражения могут быть константы, локальные переменные, глобальные и контекстные переменные, другие выражения, а также - функции. Операции выполняются над операндами в порядке их описания. Для изменения порядка выполнения операций используются вложенные выражения.

Подробнее о выражениях написано ниже.

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.

При написании выражений применяются следующие операции:

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

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

- **Унарный плюс (+)** – Не влияет на число. Преобразует строку в число.
- **Сложение (+) -** Результатом является сумма чисел.
- **Унарный минус (-)** - Результатом является число с обратным знаком. Преобразует строку в число.
- **Вычитание (-) -** Результатом является разность чисел.
- **Деление (/)** - Второй операнд воспринимается как делитель, а первый - как делимое. Результатом является частное от деления чисел.
- **Умножение (*)** – Результатом является произведение чисел.

> Для переменных типа UIDSET (список уникальных идентификаторов) 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.

- **Прибавление 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.
- **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.
- **Оператор Выбора** - Позволяет указать, какую нескольких ветвей программы выполнить в зависимости от значения указанного условия.
- **Условный оператор (IF)** - Позволяет указать, какую из двух ветвей программы выполнить в зависимости от значения (истинное или ложное) указанного условия.

Оператор Выбора и Условный оператор, используемые в выражениях, отличаются от операторов IF и CASE, являющихся отдельными конструкциями языка. Операторы для выражений имеют более простой синтаксис.

Условный оператор (Оператор IF)

‘Условный оператор’ позволяет указать, какую из двух ветвей программы выполнить в зависимости от значения (истинное или ложное) указанного условия.

Структура:

Условие - это переменная (выражение или функция), имеющая либо истинное, либо ложное значение.

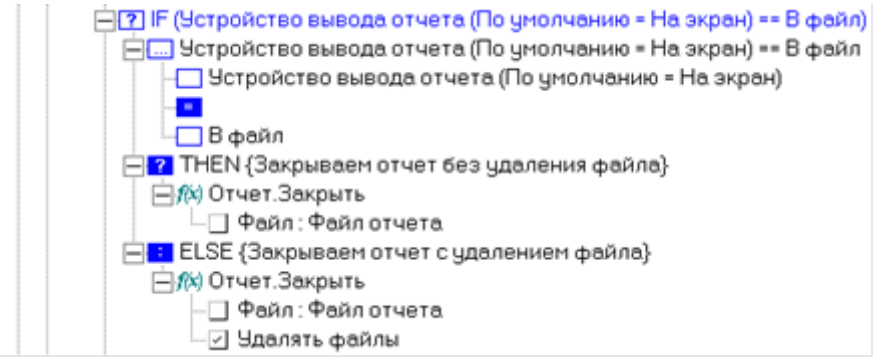
THEN - последовательность операторов данного раздела выполняется если условие имеет истинное значение.

ELSE - последовательность операторов данного раздела выполняется если условие имеет ложное значение.

Обязательное наличие сразу обоих разделов THEN и ELSE не требуется.

Для разделов THEN и ELSE рекомендуется заполнять поле ‘*Описание*’, которое будет высвечено в качестве комментария к оператору IF.

Пример использования:



Если переменная ‘Устройство вывода отчетов’ имеет значение ‘В файл’, то выполняется группа операторов, расположенная в разделе THEN. В противном случае выполняются операторы раздела ELSE.

Оператор выбора (Оператор CASE)

‘Оператор выбора’ позволяет указать, какую из нескольких ветвей программы выполнить в зависимости от значения указанного условия.

Структура:

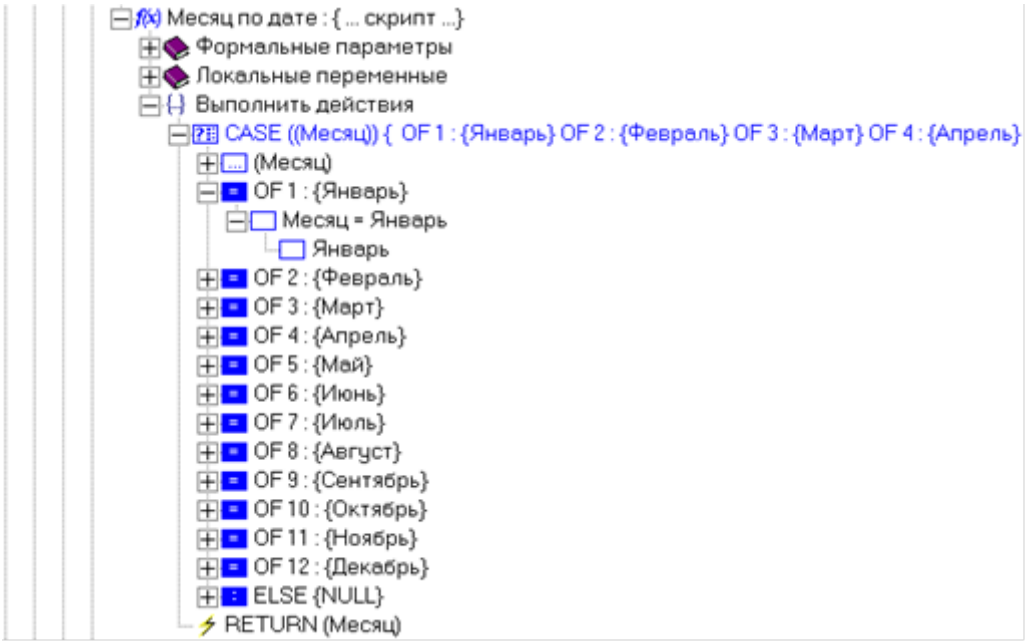
Условие - это переменная (выражение или функция), значение которой будет являться ключом для поиска подходящего варианта.

OF – значение ключа, под которым описана соответствующая ему последовательность операторов. Значения ключа последовательно сравниваются со значением условия до первого совпадения.

ELSE - Если ни одно из значений ключа не совпало со значением условием, то выполняется последовательность операторов из данного раздела.

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

Пример использования:



Функция ‘Месяц по дате’ возвращает название месяца. В условии оператора Выбора задан номер месяца. Значениями ключей являются числа от 1 до 12. Если номер месяца не совпадет ни с одним ключом (видимо на вход функции передана неверная дата), то функция возвращает значение NULL.

Оператор цикла (Оператор WHILE)

‘Оператор цикла’ обеспечивает многократное выполнение последовательности операторов

до тех пор, пока выполняется указанное условие. Если условие имеет истинное значение, то последовательность операторов будет повторена. Как только условие примет ложное значение, то управление будет передано оператору, следующему за оператором цикла.

Циклом в языках программирования называют группу операторов, выполняемых многократно. Итерацией называют один проход через цикл.

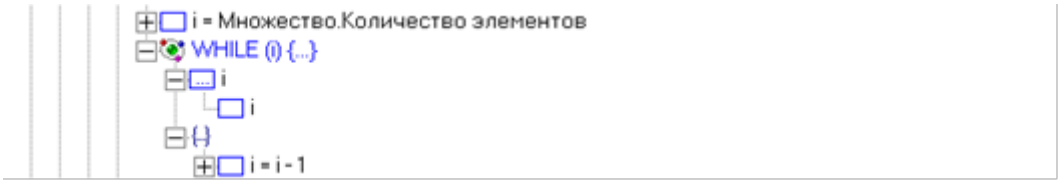
Структура

Условие - это переменная (выражение или функция), имеющая либо истинное, либо ложное значение. Условие вычисляется в начале каждой итерации.

Блок действий – Последовательность операторов программы, выполняющаяся на каждой итерации цикла.

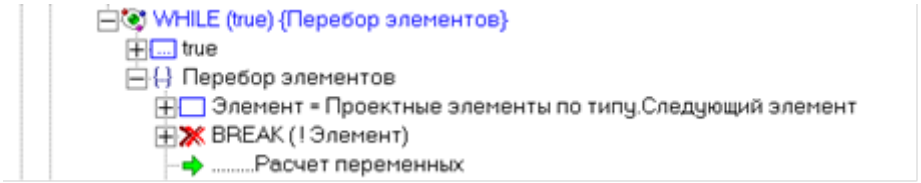
> Блок действий не выполнится ни разу, если условие будет иметь ложное значение при первом входе в цикл.

Пример использования:



Указанный цикл будет повторен i раз. Счетчик i уменьшается на 1 в начале каждой итерации.

Изменить порядок выполнения цикла можно с помощью операторов CONTINUE, BREAK, EXIT, RETURN.



Бесконечный цикл прерывается оператором BREAK.

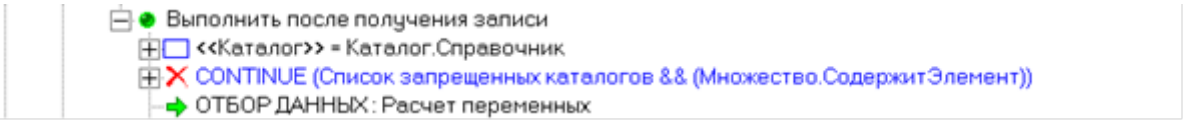
Оператор продолжить (Оператор CONTINUE)

‘Оператор продолжить’ изменяет последовательность выполнения операторов в зависимости от значения указанного условия. Если условие имеет ложное значение, то оператор CONTINUE не выполняет никаких действий. Если условие имеет истинное значение, то оператор выполняет действия в зависимости от места вызова:

- Если оператор CONTINUE вызван из основного блока процедуры (‘Выполнить действия’) или из блока ‘Всегда выполнять перед завершением’, то выполнение процедуры или блока прерывается.
- Если оператор CONTINUE вызван внутри оператора цикла, то запускается следующая итерация цикла. Выполнение текущей итерации прерывается.
- Если оператор CONTINUE вызван при отборе данных с сервера или из локальной таблицы, то вызывается считывание следующей записи.
- Если оператор CONTINUE вызван из вложенного блока действий, то считается, что данный оператор относится не к вложенному блоку, а к основным блокам процедуры (‘Выполнить действия’, ‘Всегда выполнять перед завершением’), к оператору цикла или к отбору данных.

В операторе CONTINUE не обязательно указывать условие. Такая форма оператора называется безусловной. Безусловные операторы выполняются всегда.

Пример использования:



Перебираются данные из таблицы. Оператор CONTINUE позволяет пропустить обработку запрещенных записей. Если считанная запись содержится в списке запрещенных, то оператор CONTINUE прерывает обработку текущей записи, и программа считывает следующую запись.

Оператор прервать (Оператор BREAK)

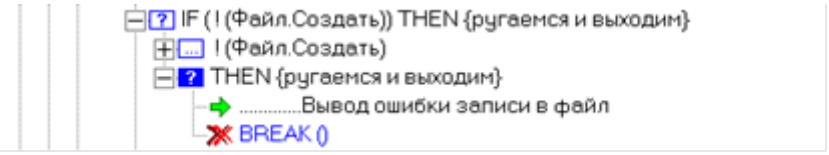
‘Оператор прервать’ изменяет последовательность выполнения операторов в зависимости от значения указанного условия. Если условие имеет ложное значение, то оператор BREAK не выполняет никаких действий. Если условие имеет истинное значение, то оператор выполняет действия в зависимости от места вызова:

- Если оператор BREAK вызван из основного блока процедуры (‘Выполнить действия’) или из блока ‘Всегда выполнять перед завершением’, то выполнение процедуры или блока прерывается.
- Если оператор BREAK вызван внутри оператора цикла, то выполнение цикла завершается, не дожидаясь выполнения условия выхода.
- Если оператор BREAK вызван при отборе данных с сервера или из локальной таблицы, то отбор данных завершается.
- Если оператор BREAK вызван из вложенного блока действий, то считается, что данный оператор относится не к вложенному блоку, а к основным блокам процедуры (‘Выполнить действия’, ‘Всегда выполнять перед завершением’), к оператору цикла или к отбору данных.

В операторе BREAK не обязательно указывать условие. Такая форма оператора называется

безусловной. Безусловные операторы выполняются всегда.

Пример использования:



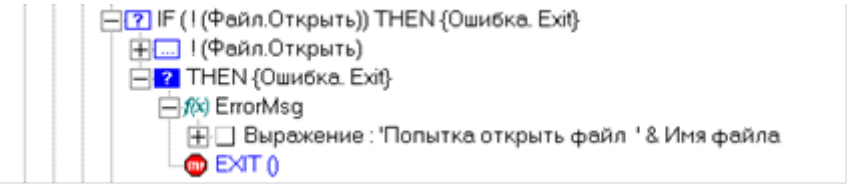
Если не удалось создать файл, то выполнение блока прерывается.

Оператор выхода (Оператор EXIT)

‘Оператор выхода’ прерывает последовательность выполнения операторов в зависимости от значения указанного условия. Если условие имеет ложное значение, то оператор EXIT не выполняет никаких действий. Если условие имеет истинное значение, то выполнение процедуры прерывается с командой возврата ’ошибка выполнения’.

В операторе EXIT не обязательно указывать условие. Такая форма оператора называется безусловной. Безусловные операторы выполняются всегда.

Пример использования:



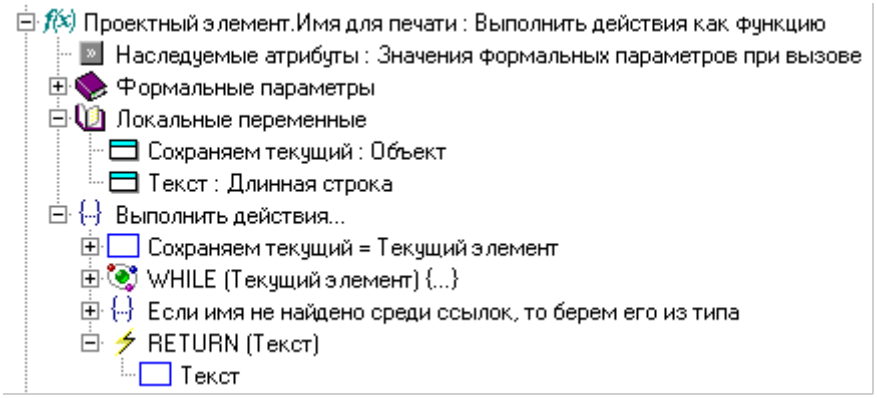
Если не удалось открыть файл, то выполнение процедуры прерывается.

Оператор возврата (Оператор RETURN)

‘Оператор возврата’ прерывает последовательность выполнения с командой возврата ’нормальное завершение’.

Если оператор RETURN используется внутри функции, то можно под оператором указать выражение. Значение указанного выражения функция вернет в качестве результата выполнения.

Пример использования в функции:

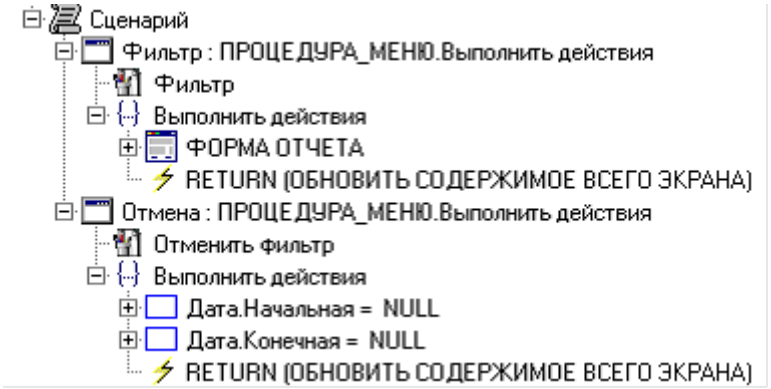


Функция возвращает строку, содержащую имя проектного элемента.

Если оператор RETURN используется внутри процедуры, то либо оператор записывается без возвращаемого результата, либо в нем явно указывается команда возврата.

Второй вариант применяется в процедурах, вызываемых из сценария вида просмотра.

Например, если по завершении процедуры требуется обновить содержимое экрана, то следует вернуть соответствующую команду.



Процедуры установки и отмены фильтров в виде просмотра возвращают команду возврата ‘Обновить содержимое экрана’.

Оператор Отложить выполнение

Оператор ‘*Отложить выполнение*’ применяется в видах просмотра при отборе данных.

Данный оператор при выполнении указанного условия временно прекращает работу блока ‘*Отбор данных*’ вида просмотра и передает управление интерфейсу (той компоненте ядра системы, которая обслуживает запущенный вид просмотра). Это дает возможность получать данные порциями и своевременно отображать их на экране, не дожидаясь завершения отбора всех данных. Интерфейс получает столько записей, сколько поместится на экране, и приостанавливает получение следующих. Последующий вызов процедуры для получения очередной порции данных продолжит выполнение блока ‘*Отбор данных*’, начиная с оператора, расположенного после оператора ‘*Отложить выполнение*’.

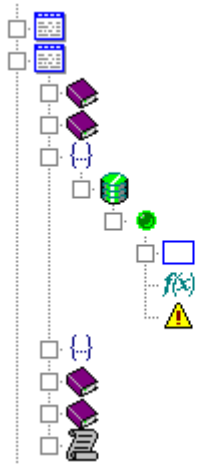
Использование оператора ‘*Отложить выполнение*’ в других местах не рекомендуется

и аналогично действию оператора RETURN.

Если условие имеет ложное значение, то оператор ‘*Отложить выполнение’* не выполняет никаких действий. Если условие имеет истинное значение, то выполнение прерывается с командой возврата ‘нормальное завершение’.

В операторе ‘*Отложить выполнение’* не обязательно указывать условие. Такая форма оператора называется безусловной. Безусловные операторы выполняются всегда.

Пример использования в виде просмотра:



При обычной схеме работы со списками данных применяются две компоненты программы. Источник данных (DATA SOURCE, DS) подготавливает записи, а интерфейс, в данном случае – это вид просмотра, отображает записи на экране. В приведенном примере DS считывает из базы данных записи документов. После считывания очередной записи управление передается интерфейсу (оператор ‘Отложить выполнение’). Интерфейс проверяет, заполнилось ли окно со списком записей. Если еще имеется место, то управление возвращается обратно в DS для получения очередной записи.

Применение оператора ‘*Отложить выполнение’* позволяет пользователю начать работу с видом просмотра, не дожидаясь считывания всего списка из базы данных. Как только считано такое количество записей, которое достаточно для заполнения окна, так можно отображать окно на экране и позволить пользователю выполнять действия с записями. Если пользователю требуются другие записи, не поместившиеся на экране, то он сообщает об этом, нажимая соответствующие клавиши (стрелка вниз, следующая страница и т.д.). Интерфейс обрабатывает полученные команды и запрашивает у DS новые записи.

Оператор Вызова процедуры или функции

‘Оператор Вызова процедуры или функции’ запускает указанную процедуру.

Если с помощью данного оператора запускается функция, то возвращаемый функцией результат игнорируется.

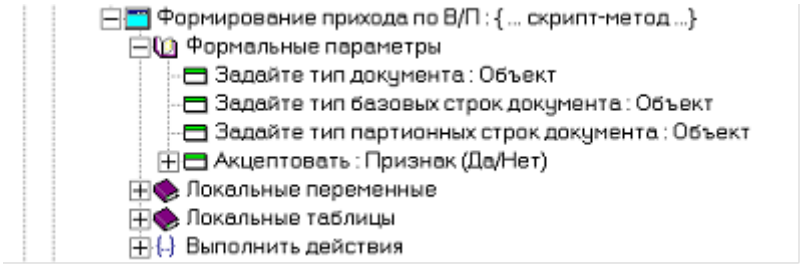
Допускается рекурсивный вызов процедур, т.е. запущенная процедура может содержать вызов самой себя.

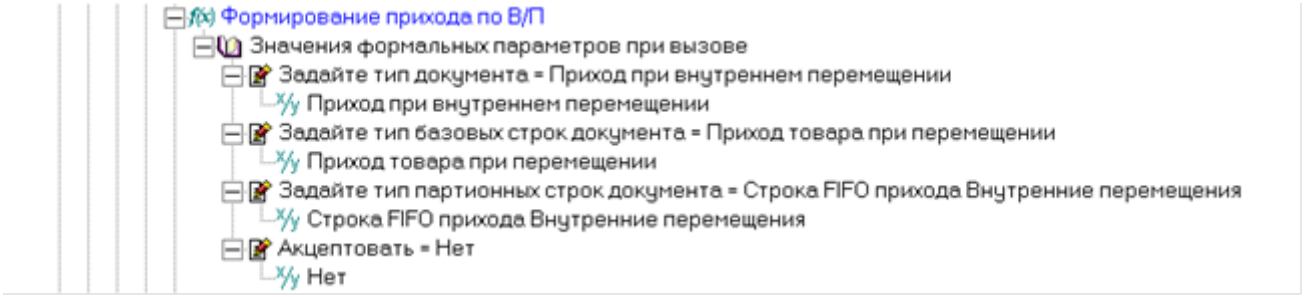
Если процедура или функция имеют формальные параметры, то необходимо указать значения этих параметров в папке ‘Значения параметров при вызове’.

Если запущенная процедура или функция завершили работу с командой возврата ’ошибка выполнения’, то соответствующее сообщение выдается в протокол.

Если для процедуры указан признак ‘Прервать при наличии ошибок’ и запущенная процедура (функция) завершилась с командой ошибки, то прерывается выполнение того блока, где данная процедура была вызвана. Это может быть либо основной блок ‘Выполнить действия’, либо блок ‘Всегда выполнять перед завершением’.

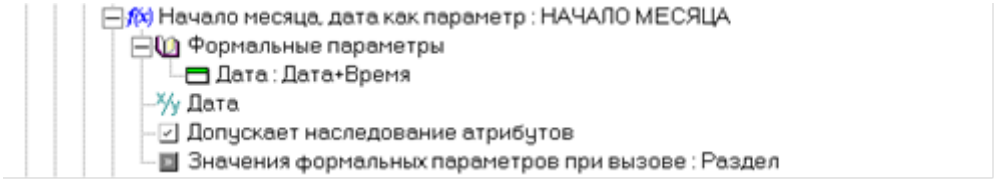
Пример вызова процедуры-метода с заполнением формальных параметров:

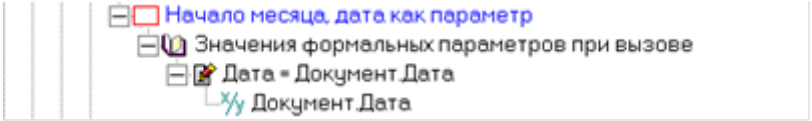




Это описание процедуры-метода.

Пример вызова функции с заполнением формальных параметров.





Это описание функции.

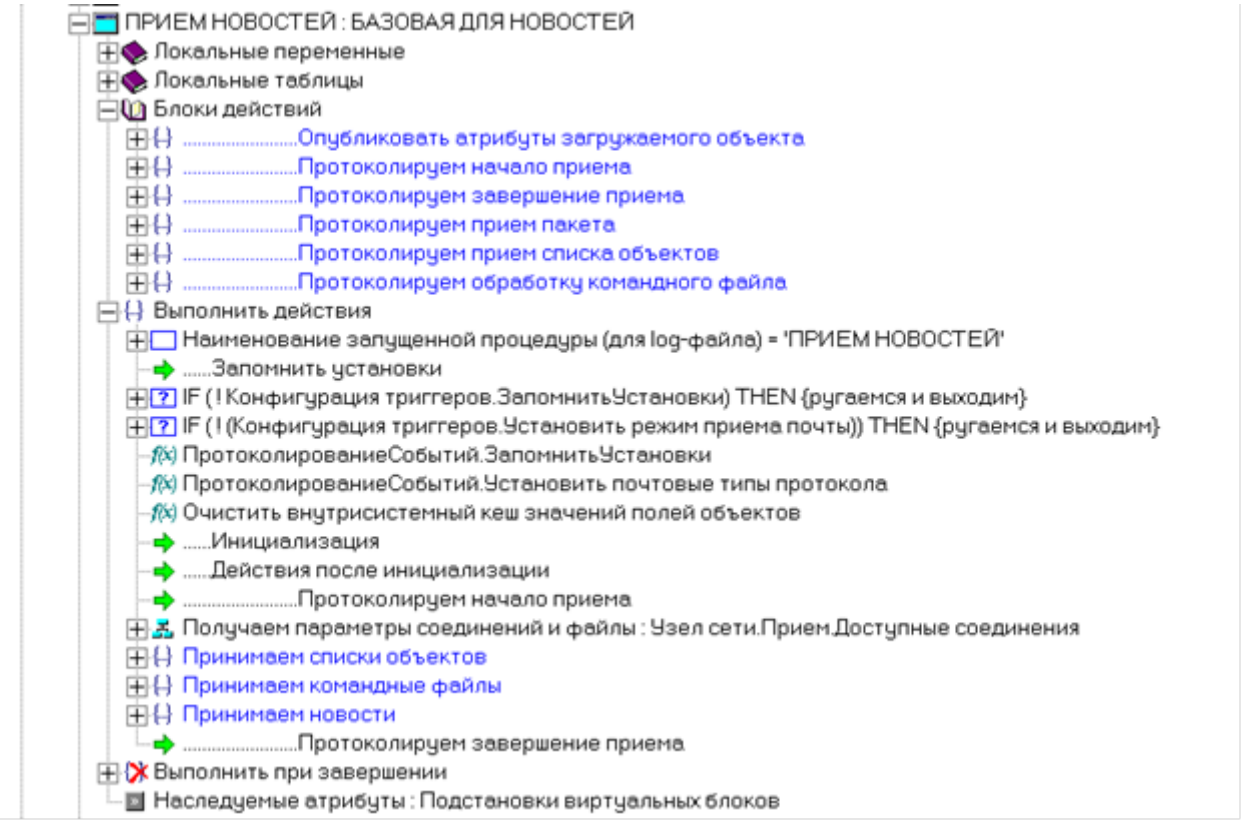
Оператор Блок

‘Оператор Блок’ применяется для группировки последовательности операторов.

Блоки могут располагаться в теле процедуры, в специальном разделе ‘*Блоки действий*’, а также в других блоках. Никаких ограничений по уровню вложенности блоков нет.

Применение блоков делает текст процедуры более структурным, и, следовательно, его легче прочитать и понять. Названия блоков удобно использовать для документирования кода.

Блоки, описанные в теле программы, выполняются автоматически по ходу выполнения ее операторов.

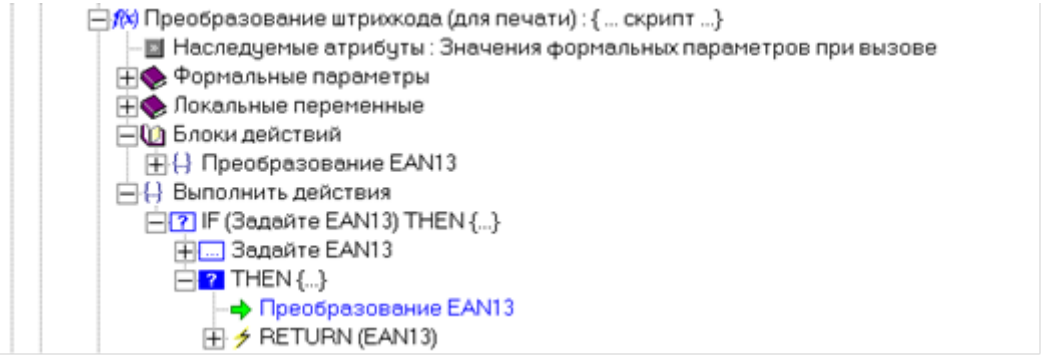


> *В данном примере разработчик расположил некоторые блоки непосредственно в теле процедуры (раздел ‘Выполнить действия’). Другие блоки он поместил в раздел ‘Блоки действий’. Часть блоков описана в базовой процедуре.*

Оператор Выполнить блок

‘Оператор Выполнить блок’ запускает указанный блок действий. В операторе можно указать либо непосредственно блок действий, либо виртуальный блок действий.

Пример использования:



В описании функции имеется блок действий ‘Преобразование EAN13’. Этот блок вызывается на выполнение в основном блоке функции ‘Выполнить действия’.

Оператор Завершить блок

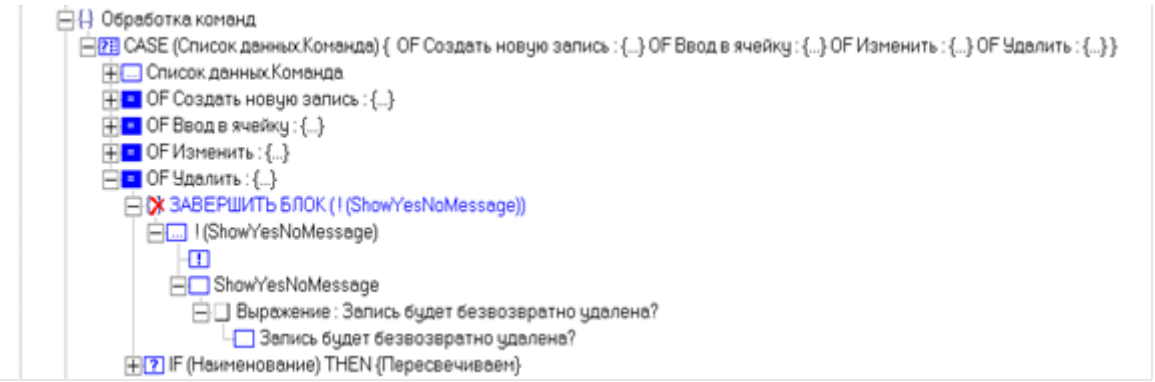
‘Оператор Завершить блок’ прерывает выполнение указанного блока в зависимости от значения указанного условия. Если условие имеет ложное значение, то данный оператор не выполняет никаких действий. Если условие имеет истинное значение, то выполнение блока прерывается.

При наличии вложенности блоков, прерывается выполнение только текущего блока. Работа того блока, из которого был запущен текущий блок, не прерывается.

В операторе Завершить блок не обязательно указывать условие. Такая форма оператора называется безусловной. Безусловные операторы выполняются всегда.

Если оператор Завершить блок применяется внутри блоков-событий в конструкциях вида ‘Отбор данных’, то данный оператор влияет только на текущий блок-событие. Текущий блок-событие прерывается. Оператор не влияет на вызов очередных блоков-событий, и отбор данных не прерывается.

Пример использования:



Оператор Завершить блок будет выполнен, если пользователь не подтвердит удаление записи.

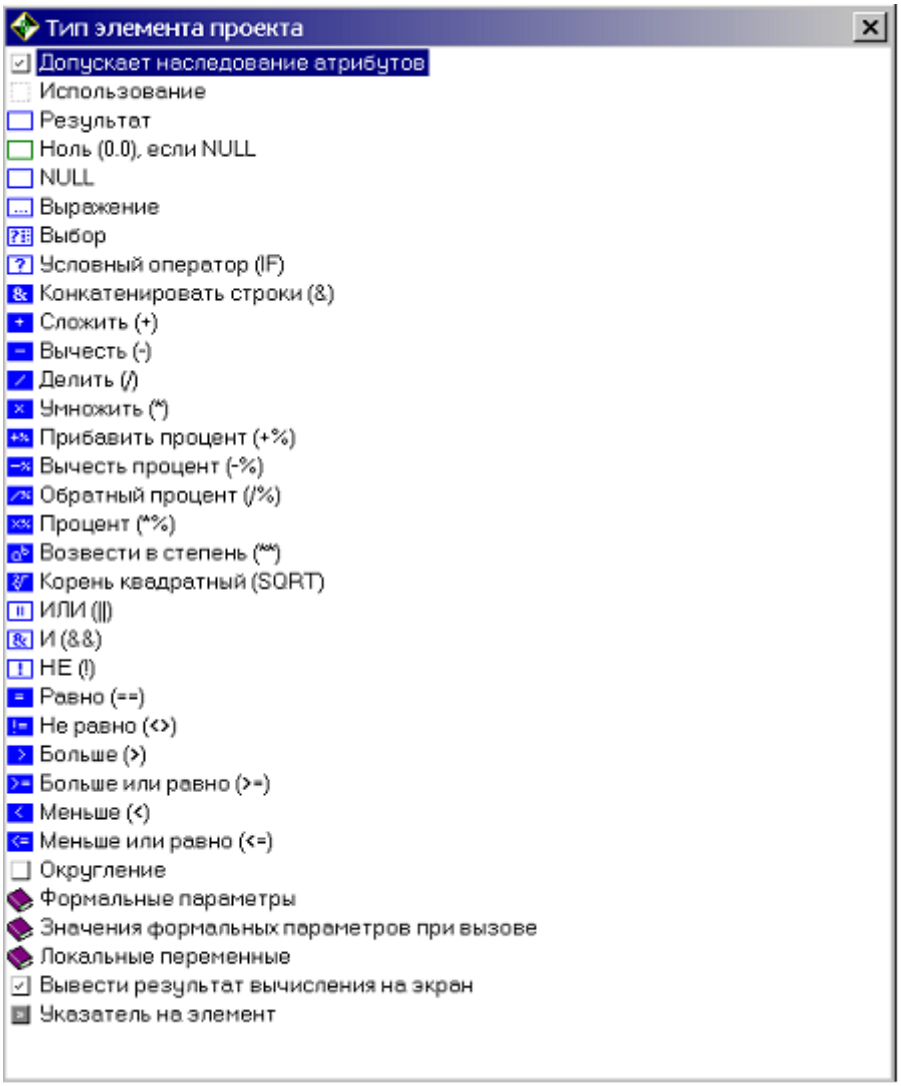
Выражения

Выражение является, наверное, самой применяемой конструкцией языка скриптов.

Выражения встречаются и в операторах присваивания, и в условиях выполнения условных и итеративных операторов, и внутри процедур и функций.

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.

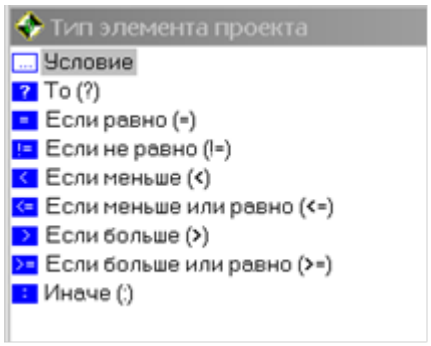
При описании выражения можно использовать следующие атрибуты:



- **Результат** - Возвращает константу, значение контекстной, локальной или глобальной переменных, результат вычисления другого выражения, функции, содержимое параметра объекта базы данных. Если результат возвращает объект, то с помощью проектного элемента ‘Уточняющий параметр’ получается значение нужного

параметра этого объекта. При необходимости можно пройтись по цепочке связанных объектов.

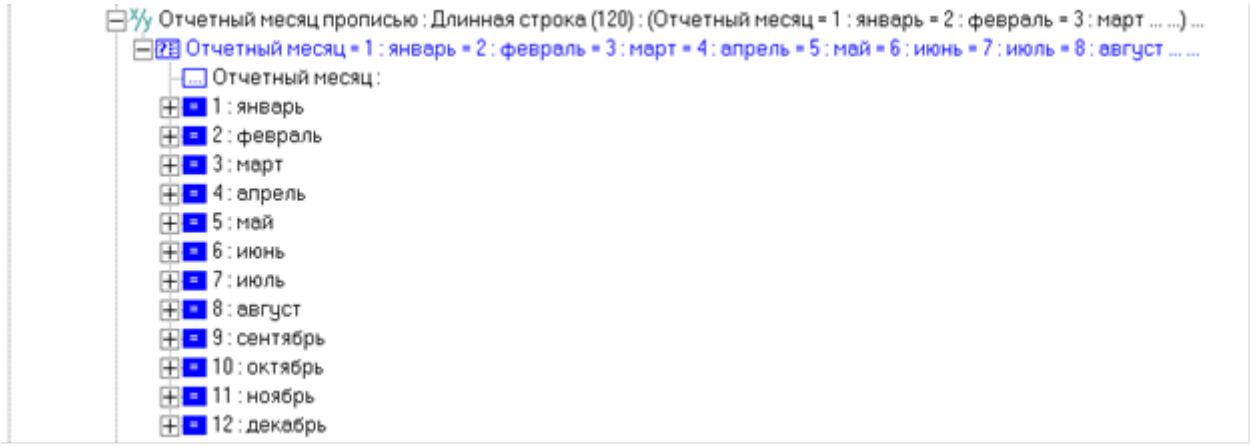
- **Ноль (0.0), если NULL** - Возвращает значение контекстной, локальной или глобальной переменных, результат вычисления другого выражения, функции, содержимое параметра объекта базы данных. Если полученный результат имеет значение NULL, то возвращает ноль (0.0).
- **NULL** – Константа NULL.
- **Выражение** - Служит для изменения порядка выполнения операторов. Применение данной операции аналогично скобкам.
- **Оператор Выбора** - Позволяет указать, какую нескольких ветвей программы выполнить в зависимости от значения указанного условия.



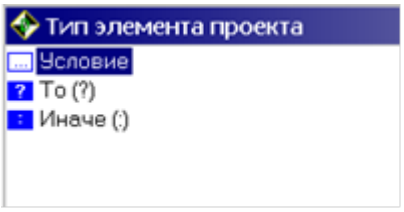
  - **Условие** - это переменная (выражение или функция), значение которой будет являться ключом для поиска подходящего варианта.
  - **То (?)** – Если условие имеет истинное значение (TRUE), то выполняется последовательность операторов этого раздела и выход из оператора выбора.
  - **Если равно (=) , 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 выполняется последовательность операторов, заданная в разделе соответствующего правила. Затем выполняется выход из оператора выбора.
  - **Иначе (:)** – Последовательность операторов данного раздела выполняется в том случае, если ни одно правило сравнения условия и значений-ключей не было выполнено.

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:



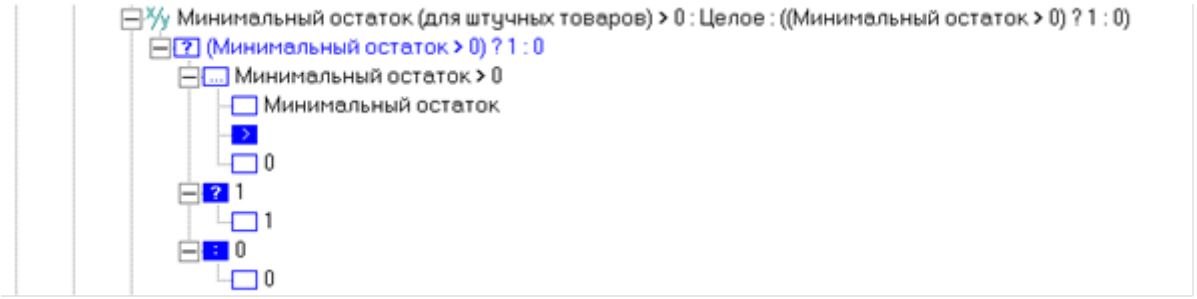
Условный оператор (IF) - Позволяет указать, какую из двух ветвей программы выполнить в зависимости от значения (истинное или ложное) указанного условия.



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

Прочие атрибуты

- **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 проверки выражений. При установленном признаке результат вычислений высвечивается на экране в отдельном окне.
- **Указатель на элемент** – применяется для указания специальных атрибутов выражения.

Версия #2

Демонов Сергей создал Thu, Mar 31, 2022 4:35 PM

Демонов Сергей обновил Thu, Mar 31, 2022 5:14 PM