ERP - система «КОМПАС»

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Раздел I

Общие правила работы

Приложение 1. ТИПЫ ДАННЫХ, ИСПОЛЬЗУЕМЫЕ В ФОРМУЛАХ И ПЕРЕЧЕНЬ СТАНДАРТНЫХ ФУНКЦИЙ

1. ТИПЫ ДАННЫХ, ИСПОЛЬЗУЕМЫЕ В ФОРМУЛАХ....2

2. ПЕРЕЧЕНЬ СТАНДАРТНЫХ ФУНКЦИЙ 5

Функции обработки текстовых строк....5

- ?S 5
- ALLTRIM() 5
- RTRIM()....6
- LEN() 6
- UPPER() Переводит символы из нижнего регистра в верхний 7
- AT() Находит фрагмент в символьной строке 7
- RAT() Находит фрагмент в символьной строке с конца 8
- SUBSTR() Выделяет подстроку из символьной строки 8
- CR(), CRN() Выделяет символ из строки 9
- LEFT() Выделяет подстроку, начиная с первого символа в строке....9
- RIGHT() Выделяет подстроку, заканчивая самым правым символом 10
- PADL() Дополняет строку до заданной длины пробелами слева....11
- PADR()Дополняет строку до заданной длины пробелами справа 11
- REPLICATE() , REPL() Повторяет строку заданное количество раз....12
- SPACE() Возвращает строку пробелов....12
- PROPER(), С_ЗАГЛ() Делает первый символ заглавной буквой 13
- STUFF() Удаляет или вставляет символы в строке 13

Функции обработки чисел 15

- ?N(<строка>,<число>) 15
- ROUND(), ОКРУГ() Округляет числовое значение 15
- ABS() Возвращает абсолютную величину арифметического значения 16

Функции обработки дат 18

1. ТИПЫ ДАННЫХ, ИСПОЛЬЗУЕМЫЕ В ФОРМУЛАХ

В формулах, создаваемых конструктором можно использовать четыре типа данных: **числовой, строковый (текст), дата и логический**.

Числовой тип позволяет задавать константы в виде:

[ws] [sn] [ddd] [.] [fmt[sn]dd]

где: [ws] - необязательные пробелы; [sn] - необязательный знак (+ или -); [ddd] - необязательные цифры; [fmt] - необязательные символы e или E (латинские); [.] - необязательная десятичная точка.

Например: +1231.19
-4564.00
12e5

Над числами можно выполнять следующие математические действия:

+ сложение;- вычитание;

* умножение;

/ деление;

% остаток от деления;

- изменение знака;

> больше;

< меньше;

>= больше или равно;

<= меньше или равно;

<> не равно.

Пример: 1024.25*2/6+(20-2)
Результат: 359.42

Заметим, что **строковые** константы (они очень часто нужны) заключаются в кавычки или апострофы, например: “константа” или ‘константа’. К строкам можно применить стандартную операцию «+». Она означает конкатенацию двух строк, т. е. «Управление» + «запасами» даст результат «Управление запасами». Кроме того, к двум строкам можно применить любые операции сравнения: **>, <, >=, <=,** !=, =. Понятие «больше» для строки означает – «дальше по алфавиту».

Тип **даты** позволяет задавать константы в виде [дата], где дата, в зависимости от настройки **«КОМПАСА»** может задаваться либо во встроенном в **«КОМПАС»** формате [дд/мм/гг] или

[дд.мм.гг], либо в формате, на который настроена операционная система. Здесь: **дд -** от одной до двух цифр номера дня, **мм** - от одной до двух цифр порядкового номера месяца, **гг** - от двух до четырех цифр года (если цифр две, то понимается XXI век). В качестве ограничителей константы даты выступают парные квадратные скобки.

Специальный вид **даты** - пустая дата, она заведомо меньше любой непустой даты. Задается пропуском всех цифр в записи константы, например [/ /].

Над датами можно выполнять следующие действия:

+ сложение даты и числа, результат - дата, увеличенная на заданное число дней;

- вычитание из даты числа, результат - дата, уменьшенная на заданное число дней;

- вычитание из даты другой даты, результат - число дней, разделяющее эти две даты;

> больше;

< меньше;

>= больше или равно;

<= меньше или равно;

<> не равно.

Пример: 1024.25*2/6+(20-2)
Результат: 359.42

Логический тип данных имеет всего две константы, которые записываются как **ИСТИНА** (*да*, или *true*, или *yes*) и **ЛОЖЬ** (*не*т, или *false*, или *no*). Над логическими данными можно совершать следующие операции: сравнивать их на равенство (=) или неравенство (<>), производить логическое умножение по И (&), логическое сложение по ИЛИ (|) и использовать одноместную операцию отрицания НЕ (!). Основное назначение логического типа данных заключается в том, что они являются результатом различных операций и функций сравнения, после чего используются в условных операторах и функции ЕСЛИ (см. раздел 2 настоящего руководства **Перечень стандартных функций**).

2. ПЕРЕЧЕНЬ СТАНДАРТНЫХ ФУНКЦИЙ

Теперь приведем перечень стандартных функций **Конструктора формул**, сгруппированный по типам аргументов.

Функции обработки текстовых строк

- **?S (<строка текст>,<строка умолчание>)**

запрос ввода с клавиатуры, возвращает строку, введенную пользователем, при этом окно ввода снабжается поясняющим текстом <строка текст>, а <строка умолчание> подается в окно ввода под корректировку;

- **ALLTRIM() Удаляет окружающие пробелы в строке символов**

Синтаксис: ALLTRIM(<символьное выражение>) -> строка символов

Аргументы:

<символьное выражение> - произвольное символьное выражение.

Результат:

ALLTRIM() возвращает строку символов с удаленными пробелами в начале и в конце строки.

Описание:

ALLTRIM() - функция работы с символьными строками, которая удаляет ведущие и завершающие пробелы из строки. Она выполняет действие сразу двух функций - LTRIM() и RTRIM(), которые удаляют из строки ведущие и заключающие пробелы соответственно. Функции ALLTRIM(), LTRIM() и RTRIM() являются противоположными по действию функциям PADR() и PADL(), которые выравнивают строки справа или слева, добавляя в них пробелы.

- **LTRIM()** Удаляет ведущие пробелы в строке символов

Синтаксис: LTRIM(<символьное выражение>) -> строка символов

Аргументы:

<символьное выражение> - произвольное символьное выражение.

Результат:

LTRIM() возвращает копию аргумента <символьная строка> с удаленными начальными пробелами. Если <символьная строка> является нулевой строкой ("") или строкой, состоящей только из пробелов, LTRIM() возвращает нулевую строку ("").

Описание:

LTRIM() является функцией обработки символьных строк, используемой для форматирования символьных строк с начальными пробелами. Ими могут быть, например, числа, преобразуемые в символьные строки функцией STR().

LTRIM() сходна с функцией RTRIM() которая удаляет конечные пробелы. Обратными к функциям ALLTRIM(), LTRIM() и RTRIM() являются функции PADR() и PADL(), которые сдвигают вправо или влево символьные строки путем добавления к ним символов-заполнителей.

- **RTRIM() Удаляет конечные пробелы в строке символов**

Синтаксис: RTRIM(<символьное выражение>) -> строка символов

Аргументы:

<символьное выражение> - произвольное символьное выражение.

Результат:

RTRIM() возвращает копию аргумента <символьная строка> с удаленными конечными пробелами. Если <символьная строка> является нулевой строкой ("") или строкой, состоящей только из пробелов, RTRIM() возвращает нулевую строку ("").

Описание:

RTRIM() является функцией обработки символьных строк, используемой для форматирования символьных строк с конечными пробелами.

RTRIM() сходна с функцией LTRIM(), которая удаляет начальные пробелы. Обратными к функциям ALLTRIM(), LTRIM() и RTRIM() являются функции PADR() и PADL(), которые сдвигают вправо или влево символьные строки путем добавления к ним символов-заполнителей.

- **LEN()** Определяет длину символьной строки

Синтаксис: LEN(<символьное выражение>) -> число

Аргументы:

<символьное выражение> - произвольное символьное выражение.

Результат:

LEN() возвращает длину символьной строки в виде целого числа. Если символьная строка пуста, то LEN() возвращает ноль.

Описание:

LEN() является функцией обработки символьных строк, которая возвращает длину символьной строки. Длина пустой строки ("") равна 0.

- **LOWER()** Переводит символы из верхнего регистра в нижний

Синтаксис: LOWER(<символьное выражение>) -> символьная строка

Аргументы:

<символьное выражение> - произвольное символьное выражение.

Результат:

LOWER() возвращает копию аргумента <символьной строки>, в которой все буквенные символы верхнего регистра преобразованы в буквенные символы нижнего регистра (прописные буквы - в строчные). Все остальные символы остаются без изменений.

- **UPPER() Переводит символы из нижнего регистра в верхний**

Синтаксис: UPPER(<символьное выражение>) -> символьная строка

Аргументы:

<символьное выражение> - произвольное символьное выражение.

Результат:

UPPER() возвращает копию аргумента <символьной строки>, в которой все буквенные символы нижнего регистра преобразованы в буквенные символы верхнего регистра (строчные буквы - в прописные). Все остальные символы остаются без изменений.

- **AT() Находит фрагмент в символьной строке**

Синтаксис: AT(<подстрока поиска>,<строка>) -> номер позиции

Аргументы:

<подстрока поиска> - символьная подстрока, которая ищется.

<строка> - символьная строка, в которой ведется поиск.

Результат:

AT() возвращает номер позиции первого появления подстроки <подстрока поиска> в строке <строка> в виде числового значения. Если подстрока <подстрока поиска> не найдена, AT() возвращает нуль.

- ### RAT() Находит фрагмент в символьной строке с конца

Синтаксис: RAT(<подстрока поиска>,<строка>) -> номер позиции

Аргументы:

<подстрока поиска> - символьная подстрока, которая ищется.

<строка> - символьная строка, в которой ведется поиск.

Результат:

RAT() возвращает номер позиции последнего появления подстроки <подстрока поиска> в строке <строка> в виде числового значения. Если подстрока <подстрока поиска> не найдена, RAT() возвращает нуль.

- ### SUBSTR() Выделяет подстроку из символьной строки

Синтаксис:

SUBSTR(<строка символов>,<начальная позиция>[,<число символов>]) -> подстрока

Аргументы:

<строка символов> - символьная строка, из которой должна быть выделена подстрока.

<начальная позиция> - начальная позиция в строке, заданной аргументом <строка символов>. Если значение этого аргумента положительное, то отсчет ведется от крайнего слева символа в аргументе <строка символов>. Если же значение аргумента - отрицательное, то от крайнего справа символа.

<число символов> - количество выделяемых символов. Если аргумент опущен, подстрока начинается со значения аргумента <начальная позиция> и продолжается до конца строки. Если значение аргумента больше числа символов, отсчитываемых от аргумента <начальная позиция> до конца строки, то излишек игнорируется.

Результат:

SUBSTR() возвращает выделенную символьную подстроку.

Описание:

SUBSTR() является функцией обработки символьных строк, которая выделяет подстроку из другой символьной строки. SUBSTR() сходна с функциями LEFT() и RIGHT(), которые выделяют подстроки, начиная с крайнего слева или крайнего справа символов аргумента <строка символов>.

- ## CR(), CRN() Выделяет символ из строки

Синтаксис:

CR(<строка символов>,<позиция>[,<заполнитель>]) -> символ

CRN(<число>,<позиция>[,<заполнитель>]) -> символ

Аргументы:

<строка символов> - символьная строка, из которой должна быть выделен символ. Для функции CRN первым аргументом является <число>, которое преобразуется к строке символов.

<позиция> - позиция символа в строке, заданной аргументом <строка символов>. Если значение этого аргумента положительное, то отсчет ведется от крайнего слева символа в аргументе <строка символов>. Если же значение аргумента - отрицательное, то от крайнего справа символа.

<заполнитель> - символ, который используется, если позиция выходит за пределы строки. Если аргумент опущен, результатом будет пустая строка.

Результат:

CR() и CRN() возвращают символ, выделенный из строки.

Описание:

CR() является функцией обработки символьных строк, которая выделяет один символ из символьной строки. CR() сходна с функцией SUBSTR(), которая выделяет подстроку. В отличие от функции SUBSTR(), функция CR() является более лаконичной и позволяет использовать заполнитель.

CRN() используется в тех случаях, когда используемая символьная строка является изображением числа.

- ## LEFT() Выделяет подстроку, начиная с первого символа в строке

Синтаксис:

LEFT(<символьная строка>, <длина подстроки>) -> подстрока

Аргументы:

<символьная строка> - символьная строка, из которой выделяется подстрока.

<длина подстроки> - длина выделяемой подстроки.

Результат:

LEFT() возвращает заданное число символов, начиная с первого символа в строке, в виде символьной подстроки. Если значение аргумента <длина подстроки>- отрицательное или ноль, LEFT() возвращает пустую строку ("""). Если значение аргумента <длина подстроки> больше длины исходной строки, LEFT() возвращает всю строку.

Описание:

LEFT() является функцией обработки символьных строк, которая возвращает подстроку, выделенную из заданной символьной строки. Она аналогична функции SUBSTR(<символьная строка>, 1, <длина подстроки>). LEFT() сходна с RIGHT(), которая возвращает подстроку, начиная с последнего символа в строке.

LEFT(), RIGHT() и SUBSTR() часто используются совместно с функциями AT() и RAT() для определения первой и/или последней позиции подстроки в строке перед ее выделением.

- **RIGHT() Выделяет подстроку, заканчивая самым правым символом**

Синтаксис:

RIGHT(<строка символов>, < длина подстроки>) -> подстрока

Аргументы:

<строка символов>**-** символьная строка, из которой выделяется подстрока.

<длина подстроки> **-** длина выделяемой подстроки.

Результат:

RIGHT() возвращает подстроку длиной, равной количеству символов, заданному в аргументе <длина подстроки>. Подстрока выбирается от конца строки, заданной в аргументе <строка символов>. Если значение аргумента <длина подстроки> отрицательно или ноль, то RIGHT() возвращает строку нулевой длины ("""). Если значение аргумента <длина подстроки> больше длины символьной строки, то RIGHT() полностью возвращает строку, заданную в аргументе <строка символов>.

Описание:

RIGHT() - функция обработки символьных строк, которая выделяет подстроку, начиная с самого правого символа аргумента <строка символов>. Ее действие аналогично действию выражения SUBSTR (<строка символов>,-<число символов>). Например, RIGHT("ABC", 1) - это то же самое, что и SUBSTR("ABC", -1).

Функция RIGHT() похожа на функцию LEFT(), которая извлекает подстроку, начиная с самого левого символа аргумента <строка символов>.

Функции RIGHT(), LEFT() и SUBSTR() часто используют в сочетании с функциями AT() и RAT(), определяющими первую и/или последнюю позицию подстроки до ее извлечения.

- ### PADL() Дополняет строку до заданной длины пробелами слева

Синтаксис: PADL(<символьная строка>, <длина строки>) -> строка

Аргументы:

<символьная строка> - символьная строка, подлежащая преобразованию.

<длина строки> - длина результирующей строки.

Результат:

PADL() возвращает строку указанной длины, используя исходную строку и, возможно, некоторое количество пробелов слева от нее. Если <длина строки> меньше длины исходной строки, то функция PADL() действует аналогично функции LEFT().

Описание:

PADL() является функцией обработки символьных строк, которая возвращает подстроку, полученную из заданной символьной строки путем дополнения пробелами слева, если длина исходной строки недостаточна, либо обрезанием строки справа, если длина строки излишне велика. PADL() сходна с PADR(), которая отличается тем, что дополняет пробелы справа.

- ### PADR()Дополняет строку до заданной длины пробелами справа

Синтаксис: PADR(<символьная строка>, <длина строки>) -> строка

Аргументы:

<символьная строка> - символьная строка, подлежащая преобразованию.

<длина строки> - длина результирующей строки.

Результат:

PADR() возвращает строку указанной длины, используя исходную строку и, возможно, некоторое количество пробелов справа от нее. Если <длина строки> меньше длины исходной строки, то функция PADR() действует аналогично функции LEFT().

Описание:

PADR() является функцией обработки символьных строк, которая возвращает подстроку, полученную из заданной символьной строки путем дополнения пробелами справа, если длина исходной строки недостаточна, либо обрезанием строки справа, если длина строки излишне велика. PADR() сходна с PADL(), которая отличается тем, что дополняет пробелы слева.

- ### REPLICATE() , REPL() Повторяет строку заданное количество раз

Синтаксис:

REPLICATE(<строка символов>, <число повторений>) -> строка символов

Аргументы:

<строка символов> - повторяемая символьная строка.

<число повторений> - количество повторений строки.

Результат:

REPLICATE() возвращает символьную строку. При количестве повторений, равном нулю, функция возвращает строку нулевой длины ("").

Описание:

REPLICATE() - это функция обработки символьных строк, которая используется для многократной выдачи на экран, вывода на печать одним или более символами. REPLICATE() подобна SPACE(), возвращающей заданное количество пробелов.

- ### SPACE() Возвращает строку пробелов

Синтаксис: SPACE(<число пробелов>) -> строка пробелов

Аргументы:

<число пробелов> - число пробелов для построения строки.

Результат:

SPACE() возвращает символьную строку. Если значение аргумента <число пробелов> равно нулю, SPACE() возвращает строку нулевой длины ("").

Описание:

SPACE() является функцией обработки символьных строк, используемой для возврата строки, состоящей из заданного числа пробелов. Ее действие подобно действию функции REPLICATE(" ",<количество символов>).

- **PROPER(), С_ЗАГЛ() Делает первый символ заглавной буквой**

Синтаксис: PROPER(<символьная строка>) -> символьная строка

С_ЗАГЛ(<символьная строка>) -> символьная строка

Аргументы:

<символьная строка> **-** произвольная символьная строка.

Результат:

PROPER() возвращает копию аргумента <символьной строки>, в которой первый буквенный символ нижнего регистра преобразован в буквенный символ верхнего регистра, а оставшиеся буквенные символы верхнего регистра преобразованы в буквенные символы нижнего регистра. Все остальные символы остаются без изменений.

- **STUFF() Удаляет или вставляет символы в строке**

Синтаксис:

STUFF(<строка символов>,<начало замены>, <число удаляемых символов>,<вставляемая подстрока>) -> строка символов

Аргументы:

<строка символов> - символьная строка, в которую вставляются или из которой удаляются символы.

<начало замены> - начальная позиция в строке, с которой происходит вставка (удаление).

<число удаляемых символов> - число символов, подлежащих удалению.

<вставляемая подстрока> - вставляемая строка.

Результат:

STUFF() возвращает копию аргумента <строка символов> с удаленными символами и вставленной строкой, значение которой задано аргументом <вставляемая подстрока>.

Описание:

STUFF() является функцией обработки символьных строк, удаляющей символы, количество которых задано в аргументе <число удаляемых символов>, из аргумента <строка символов>, начиная с позиции, заданной аргументом <начало замены>. Затем она вставляет значение аргумента <вставляемая подстрока> в полученную строку, начиная с символа, номер которого

определен в аргументе <начало замены>, и формирует возвращаемую строку. При этом STUFF() может выполнять шесть следующих операций:

Вставка. Если значение аргумента <число удаляемых символов> равно нулю, из аргумента <строка символов> символы не удаляются. Значение аргумента <вставляемая подстрока> вставляется, начиная с позиции, заданной аргументом <начало замены>, после чего строка возвращается.

Пример: STUFF("My dog has fleas",12,0,"no ")

Результат: "My dog has no fleas"

Замена. Если значение аргумента <вставляемая подстрока> имеет ту же длину, что и значение аргумента <число удаляемых символов>, значение аргумента <вставляемая строка> заменяет символы, начиная с позиции, заданной аргументом <начало замены>. Удаляется то же число символов, что и вставляется, и полученная строка имеет такую же длину, что и первоначальная.

Пример: STUFF("My dog has fleas",12,5,"bones")

Результат: "My dog has bones"

Удаление. Если значение аргумента <вставляемая подстрока> является строкой с нулевой длиной (""), из строки, заданной аргументом <строка символов>, удаляется число символов, заданное аргументом <число удаляемых символов>, и строка возвращается без каких-либо добавленных символов.

Пример: STUFF("My dog has fleas",1,3,"")

Результат: "dog has fleas"

Замена и вставка. Если значение аргумента <вставляемая подстрока> больше, чем значение аргумента <число удаляемых символов>, все символы, начиная с позиции, установленной аргументом <начало замены>, в соответствии со значением аргумента <число удаляемых символов>, удаляются, а затем значение аргумента <вставляемая подстрока> вставляется в строку. Поскольку вставляется больше символов, чем удаляется, полученная строка всегда длиннее первоначальной.

Пример: STUFF("My dog has fleas",8,3,"does not have")

Результат: "My dog does not have fleas"

Замена и удаление. Если длина аргумента <вставляемая подстрока> меньше, чем значение аргумента <число удаляемых символов>,лишние символы будут удалены, а затем произойдет вставка. Результирующая строка при этом будет короче исходной.

Пример: STUFF("My dog has fleas",8,3,"is")

Результат: "My dog is fleas"

Замена и удаление оставшихся символов. Если значение аргумента <число удаляемых символов> больше или равно числу оставшихся символов, начиная с позиции ,заданной аргументом <начало замены>, аргумента <строка символов>, все эти оставшиеся символы удаляются перед вставкой значения аргумента <вставляемая подстрока>.

Пример: STUFF("My dog has fleas",8,10,"is")

Результат: "My dog is"

Функции обработки чисел

- **?N(<строка>,<число>)**

запрос ввода с клавиатуры, возвращает число, введенное пользователем, при этом окно ввода снабжается поясняющим текстом <строка>, а <число> подается в окно ввода под корректировку.

- **ROUND(), ОКРУГ() Округляет числовое значение**

Синтаксис:

ROUND(<число>, <дробная часть>) -> округленное число

Аргументы:

<число> - округляемое число.

<дробная часть> - определяет количество десятичных разрядов, до которых надо округлить. Отрицательное значение аргумента <дробная часть> приводит к округлению целой части числа.

Результат:

ROUND() возвращает округленное число.

Описание:

ROUND() - арифметическая функция, которая округляет число до количества десятичных разрядов, заданных аргументом <дробная часть>.

Отрицательного значения приводит к округлению целой части числа. Отрицательное значение аргумента <дробная часть> указывает количество десятичных разрядов слева от запятой, до которых надо округлить число.

Числа от 5 до 9 включительно округляются в большую сторону. Числа меньше пяти - в меньшую.

- **ABS() Возвращает абсолютную величину арифметического значения**

Синтаксис:

ABS(<арифметическое выражение>) -> неотрицательное число

Аргументы:

<арифметическое выражение> **-** исходное арифметическое выражение.

Результат:

ABS() возвращает число, представляющее абсолютное значение ее аргумента. Возвращаемое значение - это положительное число или ноль.

Описание:

ABS() - это числовая функция, которая служит для определения величины числового значения независимо от его знака. Это позволяет, например, определить разность между двумя числами в виде положительного значения, не зная заранее, какое из них больше.

Формально функция ABS(X) определяется в терминах ее аргумента X следующим образом: если X > 0, то ABS(X) возвращает X, иначе ABS(X) возвращает -X.

- **STR() Преобразует числовое значение в символьную строку**

Синтаксис:

STR(<число>[,<длина строки>[,<дробная часть>]]) -> число в виде строки

Аргументы:

<число> **-** числовое выражение, подлежащее преобразованию в строку символов.

<длина строки> **-** длина возвращаемой символьной строки, включая дробную часть, десятичную точку и знак.

<дробная часть> **-** число возвращаемых разрядов дробной части.

Результат:

STR() возвращает значение аргумента <число>, преобразованное в строку символов. Если необязательные аргументы длины и величины дробной части не заданы, STR() возвращает символьную строку в соответствии со следующими правилами:

Выражение	Длина возвращаемого значения
Выражения/константы	Минимум 10 цифр плюс дробная часть
VAL()	Минимум три цифры
MONTH() или DAY()	Три цифры
YEAR()	Пять цифр

Описание:

STR() является арифметической функцией, которая преобразует числовые значения в символьные строки. Она обычно используется при конкатенации числовых значений с символьными строками. STR() применяется при вывода на дисплей чисел и пр.

Обратной для STR() является функция VAL(), которая преобразует строки символов в целые числа.

Примечания:

1. Если значение аргумента <длина строки> меньше, общее число цифр, заданное аргументом <число>, STR() возвращает звездочки (******) вместо изображения числа.
2. Если значение аргумента <длина строки> меньше, чем величина дробной части, число округляется до имеющегося количества разрядов дробной части.
3. Если аргумент <длина строки> задан, а аргумент <дробная часть> опущен (нет дробной части), возвращаемое значение округляется до целого числа.

- **VAL() Преобразует число в символьной форме в числовой тип**

Синтаксис:

VAL (<число в символьной форме>) -> число

Аргументы:

<число в символьной форме> - преобразуемое символьное выражение.

Результат:

VAL() возвращает значение аргумента <число в символьной форме> преобразованное в числовое значение, включая дробную часть.

Описание:

VAL() - функция, которая преобразует строку символов, содержащую цифры, в числовое значение. Функция VAL() просматривает значение аргумента <число в символьной форме> до тех пор, пока не встретится вторая десятичная точка, нечисловой символ или конец выражения. Лидирующие пробелы игнорируются. VAL() возвращает число десятичных знаков такое же, как и в аргументе <число в символьной форме>.

VAL() - функция, обратная STR(), которая преобразует числовые значения в символьные строки.

Функции обработки дат

- ?D(<строка>,<дата>) запрос ввода с клавиатуры

возвращает дату, введенную пользователем,

при этом окно ввода снабжается поясняющим текстом <строка>, а <дата> подается в окно ввода под корректировку.

- **DATE() Возвращает текущую календарную дату**

Синтаксис: **DATE() -> текущая дата**

Аргументы: нет.

Результат:

DATE() возвращает текущую дату в виде значения типа даты.

Описание:

DATE() - функция, которая дает возможность получить текущую дату для того, чтобы сравнить другие даты с текущей датой или произвести арифметические вычисления над датами с участием текущей даты.

- **TIME(), ВРЕМЯ() Возвращает текущее время**

Синтаксис: **TIME() -> текущее время**

ВРЕМЯ() -> текущее время

Аргументы: нет.

Результат:

TIME() и ВРЕМЯ() возвращают текущее время в виде символьной строки.

Описание:

TIME() и ВРЕМЯ() - функции, которые дают возможность получить текущее время для того, чтобы сравнить другое время с текущим временем. Функции возвращают текущее время в формате “чч:мм:сс”.

- ### DAY() Возвращает номер дня в виде числа

Синтаксис: DAY(<дата>) -> номер дня

Аргументы:

<дата> **-** подлежащая преобразованию дата.

Результат:

DAY() возвращает целое число в пределах от нуля до 31. Если аргумент пустой, DAY() возвращает 0.

Описание:

DAY() - функция, которая используется для преобразования значения даты в номер дня в месяце. Она используется в комбинации с функциями CMONTH() и YEAR() для форматирования дат, а также при различных вычислениях над датами.

- ### DOW() Преобразует дату в числовое значение дня недели

Синтаксис: DOW(<дата>) -> номер дня

Аргументы:

<дата> **-** значение даты, подлежащее преобразованию.

Результат:

DOW() возвращает номер дня недели в виде числа в пределах от одного до семи. Первый день недели - это воскресенье, последний - суббота. Если <дата> имеет пустое значение, DOW() возвращает нуль.

Описание:

DOW() - это функция преобразования даты, которая преобразует значение даты в число, идентифицирующее день недели. Функция полезна при выполнении расчетов над днями недели. Функция DOW() похожа на функцию CDOW(), которая возвращает день недели в виде строки символов вместо числа.

- ### DTOC() Преобразует дату в символьную строку

Синтаксис: DTOC(<дата>) -> строка-дата

Аргументы:

<дата> **-** это дата, значение которой преобразуется.

Результат:

DTOC() возвращает дату, представленную в виде строки символов. Возвращаемое значение форматируется с учетом установленного текущего формата даты. Нулевая дата преобразуется в строку пробелов, равную длине, принятой для текущего формата

Описание:

DTOC() - функция преобразования даты, которая используется, когда нужно выделить дату в формате, установленном в конфигурации пакета, в качестве элемента символьного выражения.

- ### MONTH() Определяет по дате номер месяца

Синтаксис: MONTH(<дата>) -> номер месяца

Аргументы:

<дата> **-** значение даты.

Результат:

MONTH() возвращает целое число, которое находится в интервале от 0 до 12. Ноль получается при обработке нулевой даты (CTOD("")).

Описание:

MONTH() - функция обработки даты, которая может быть полезна, если в процессе вычислений необходимо числовое значение месяца. MONTH() входит в группу функций, которые возвращают компоненты даты как целое число. В эту группу также входят функции DAY() и YEAR(), которые возвращают числовые значения дня и года. Функция CMONTH() позволяет по значению даты определить название месяца.

- ### YEAR() Преобразует дату в номер года в числовом виде

Синтаксис: YEAR (<дата>) -> год

Аргументы:

<дата> **-** значение даты.

Результат:

YEAR() возвращает номер года, заданного значением даты в виде четырехзначного числового значения. Заданная нулевая дата (CTOD("")) возвращает ноль.

Описание:

YEAR() - функция преобразования даты, которая используется для перевода заданной даты в числовое значение года. Функция может быть использована при вычислениях или для форматирования дат.

YEAR() входит в группу функций, которые возвращают компоненты дат в виде чисел. К этой группе относятся также функции DAY() и MONTH(), которые возвращают значения дня и месяца в виде числовых значений.

- ### CDOW() Преобразует дату в название дня недели

Синтаксис: CDOW(<дата>) -> название дня недели

Аргументы:

<дата> **-** значение даты.

Результат:

CDOW() возвращает название дня недели в виде символьной строки. Первая буква - заглавная (в верхнем регистре), остальные - строчные (в нижнем регистре). Для нулевой даты CDOW() возвращает пустую строку ("").

Описание:

CDOW() - это функция преобразования, которая используется для форматированной выдачи даты.

- ### CMONTH() Преобразует дату в название месяца

Синтаксис: CMONTH(<дата>) -> название месяца

Аргументы:

<дата> **-** дата для преобразования.

Результат:

CMONTH() возвращает название месяца по заданной дате в виде строки символов. Первый символ названия месяца - заглавный (в верхнем регистре), остальные - строчные (в нижнем регистре). Для нулевой даты CMONTH() возвращает пустую строку ("").

Описание:

CMONTH() - функция преобразования дат, которая предназначена для создания форматированных дат, представленных в виде символьных строк, для отчетов, меток, при выдаче на экран.

MONTH(<дата>) - возвращает число, равное номеру месяца в дата.

YEAR(<дата>) - возвращает число, равное номеру года в <дата>.

Прочие функции:

- **MIN() Возвращает меньшее из двух чисел, дат или строк**

Синтаксис: **MIN(<число1>,<число2>) -> меньшее число**

MIN(<дата 1>,<дата 2>) -> меньшая дата

MIN(<строка 1>,<строка 2>) -> меньшая строка

Аргументы:

<число1>, <число2> - сравниваемые числовые значения.

<дата 1>, <дата 2> - сравниваемые значения дат.

<строка1>, <строка2> - сравниваемые строки.

Результат:

MIN() возвращает наименьший из двух аргументов. Возвращаемое значение того же типа данных, что и аргументы.

Описание:

MIN() - функция обработки чисел, дат и строк, используемая для выделения значения выражения меньшего, чем заданный минимум. Обратной функцией для MIN()является MAX(), которая возвращает большее из двух числовых значений, дат или строк.

- **MAX() Возвращает большее из двух чисел, дат или строк**

Синтаксис: **MAX(<число1>,<число2>) -> большее число**

MAX(<дата 1>,<дата 2>) -> большая дата

MAX(<строка 1>,<строка 2>) -> большая строка

Аргументы:

<число1>, <число2> - сравниваемые числовые значения.

<дата 1>, <дата 2> - сравниваемые значения дат.

<строка1>, <строка2> - сравниваемые строки.

Результат:

MAX() возвращает наибольший из двух аргументов. Возвращаемое значение того же типа данных, что и аргументы.

Описание:

MAX() - функция обработки чисел, дат и строк, используемая для выделения значения выражения большего, чем заданный минимум. Обратной функцией для MAX() является MIN(), которая возвращает меньшее из двух числовых значений, дат или строк.

- ### EMPTY(), ПУСТО() Определяет, является ли значение пустым

Синтаксис: **EMPTY(<выражение>) -> логическое значение**

ПУСТО(<выражение>) -> логическое значение

Аргументы:

<выражение> - выражение любого типа, кроме логического.

Результат:

EMPTY() возвращает значение **Истина**, если результат выражения имеет пустое значение, в противном случае она возвращает **Ложь**. Критерии, по которым определяется пустое значение, зависят от типа данных выражения <выражение> в соответствии со следующими правилами:

Тип данных	**Критерий пустоты**
символ	пробел или пустая строка
число	0
дата	пустая дата: [/ /] или CTOD("")

Описание:

Функция EMPTY() предназначена для определения, является ли значение пустым.

- ### IIF(), IF(), ЕСЛИ() Вычисляет одно из двух выражений

Синтаксис:

IIF(<условие>, <выражение 1>, <выражение 2>) -> вычисленное выражение

IF(<условие>, <выражение 1>, <выражение 2>) -> вычисленное выражение

ЕСЛИ(<условие>, <выражение 1>, <выражение 2>) -> вычисленное выражение

Аргументы:

<условие> **-** это логическое выражение, управляющее выбором выражения для расчета.

<выражение 1> **-** выражение любого типа, значение которого возвращается, если <условие> - **Истина**.

<выражение 2> **-** выражение любого типа, значение которого возвращается, если <условие> - **Ложь**. Этот аргумент обязательно должен быть того же типа, что и <выражение 1>.

Результат:

IIF() возвращает результат вычисления аргумента <выражение 1>, если <условие> оценивается как **Истина**, или аргумента <выражение2>, если <условие> оценено как **Ложь**.

Тип данных возвращаемого значения совпадает с типом данных двух последних аргументов.

Описание:

IIF() - это логическая функция преобразования. Она обеспечивает механизм оценки условия внутри выражения. В зависимости от логического условия, в выражении будет использовано одно из двух заданных значений.