

айтида®
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Оглавление

КРАТКОЕ ОПИСАНИЕ ЯЗЫКА ВЫЧИСЛИТЕЛЯ СКРИПТОВ АЙТИДА 2
ОСНОВНЫЕ ПОЛОЖЕНИЯ 2
Набор символов 3
Константы 5
Идентификаторы 8
Понятия 8
Ключевые слова 9
Блок команд 9
Ключевые слова 10
Слово IF 10
Слово WHILE (ПОКА) 10
Слово *BREAK* 11
Слово *CONTINUE* 11
Слово *RETURN* 12
Слово *TRY/CATCH* 12
Слово *PUBLIC* 13
Слово *__SQL* 14
Комментарии 15
Лексемы 16
ФУНКЦИИ 17
Введение 17
Определения функций 17
Вызовы функций 17
ПРИМЕРЫ СКРИПТОВ 18
Текст обработки загрузки курсов валют 18

Скрипт автоматического создания документа Выпуск и комплектация при записи Документа кассовой смены 20
Скрипт автоматического создания документа Выписка из банка, на основании отмеченных в журнале платежных поручений 22
ВСТРОЕННЫЕ ФУНКЦИИ ВЫЧИСЛИТЕЛЯ 24
ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ 24
ФУНКЦИИ ДЛЯ РАБОТЫ С ДАТАМИ 32
ФУНКЦИИ ДЛЯ РАБОТЫ С ЧИСЛАМИ 36
ЛОГИЧЕСКИЕ ФУНКЦИИ 37
ФУНКЦИИ ДЛЯ РАБОТЫ С СОЕДИНЕНИЯМИ ODBC 38
ФУНКЦИИ ДЛЯ РАБОТЫ С ДАННЫМИ 39
ФУНКЦИИ ДЛЯ РАБОТЫ С КОНТЕКСТАМИ 42
ФУНКЦИИ ДЛЯ РАБОТЫ С ТАБЛИЦАМИ 49
СИСТЕМНЫЕ ФУНКЦИИ 56
ФУНКЦИИ ДЛЯ РАБОТЫ С СОКЕТАМИ 64
ФУНКЦИИ ДЛЯ РАБОТЫ С FTP СЕРВЕРАМИ 65
ФУНКЦИИ ДЛЯ РАБОТЫ С HTTP СЕРВЕРАМИ 68
ФУНКЦИИ ДЛЯ РАБОТЫ С COM ПОРТОМ 71
ФУНКЦИИ ДЛЯ РАБОТЫ С ФАЙЛАМИ 73
ФУНКЦИИ ДЛЯ ФОРМИРОВАНИЯ XML ТЕКСТОВ 78
ФУНКЦИИ ДЛЯ РАБОТЫ С ZIP АРХИВАМИ 81
ФУНКЦИИ ДЛЯ РАБОТЫ С RAR АРХИВАМИ 84
ФУНКЦИИ ДЛЯ РАБОТЫ С ОТЛАДЧИКОМ 87
КЛЮЧЕВЫЕ СЛОВА 87

КРАТКОЕ ОПИСАНИЕ ЯЗЫКА ВЫЧИСЛИТЕЛЯ СКРИПТОВ АЙТИДА

ОСНОВНЫЕ ПОЛОЖЕНИЯ

Для обеспечения максимальной гибкости настройки система Айтида во многих модулях поддерживает ввод и выполнение пользовательских скриптов, которые позволяют значительно расширить стандартный функционал системы, адаптировав его под конкретные потребности конечного клиента. Выполнением таких скриптов занимается модуль системы, называемый Вычислитель. Основной задачей Вычислителя является выполнение программ (скриптов), написанных на специальном языке. Отличительной особенностью этого языка является то, что он очень тесно

взаимодействует с данными в базе данных посредством Понятий. В остальном данный язык является простым функциональным языком, оптимизированным под вычисление различных выражений.

Набор символов

Набор символов Вычислителя состоит из букв, цифр, символа @ и символов пунктуации, которые имеют специальное значение. Вы строите программу в Вычислителе, комбинируя символы в осмысленные операторы.

Буквы, цифры и подчеркивание

Набор символов Вычислителя содержит прописные и строчные буквы латинского и русского алфавитов, 10 десятичных цифр арабской системы исчисления, символ подчеркивания (_) и символ @.

- Прописные английские буквы:

  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- Строчные английские буквы:

  a b c d e f g h i j k l m n o p q r s t u v w x y z

- Прописные русские буквы:

  А Б В Г Д Е Ё Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я

- Строчные русские буквы:

  а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я

- Десятичные цифры:

  0 1 2 3 4 5 6 7 8 9

- Символ подчеркивания (_)
- Символ @

Эти символы используются для формирования понятий, констант, идентификаторов и ключевых слов, описанных далее.

Интерпретатор Вычислителя обрабатывает прописные и строчные буквы, как одинаковые символы. Например, если в идентификаторе использована строчная буква а, то Вы можете заменить ее на прописную букву А.

Разделительные символы

Пробел, смена строки, возврат каретки и символ новой строки называются разделительными символами, т.к. они выполняют функцию пробелов между словами и строками в напечатанной странице. Эти символы отделяют задаваемые пользователем элементы, например, константы и идентификаторы, от других элементов программы.

Интерпретатор Вычислителя игнорирует разделительные символы, если они не служат для целей разделения или не являются компонентами символьных констант или строковых литералов. Таким образом, можно использовать дополнительные разделительные символы для улучшения восприятия программы. Интерпретатор рассматривает комментарии, как разделительные символы. (Комментарии рассматриваются в разделе "Комментарии".)

Знаки пунктуации и специальные символы

Знаки пунктуации и специальные символы из набора символов имеют самое разное предназначение, от организации текста программы до определения задач, которые будут выполнены программой. В Таблице 1 приведен список знаков пунктуации и специальных символов из набора символов Вычислителя.

Таблица 1. Знаки пунктуации и специальные символы

Символ	Название	Символ	Название
,	запятая	{	левая фигурная скобка
.	точка	}	правая фигурная скобка
;	точка с запятой	<	левая угловая скобка
:	двоеточие	>	правая угловая скобка
?	знак вопроса	!	восклицательный знак
'	одинарная цитатная скобка	/	знак деления
"	двойная цитатная скобка	+	плюс
(	левая круглая скобка	%	процент
)	правая круглая скобка	*	звездочка
[	левая прямоугольная скобка	-	минус
]	правая прямоугольная скобка	=	равно

Все эти символы имеют специальное значение. В данном руководстве описывается их использование. Любой знак пунктуации, не приведенный в Таблице 1, может быть использован только в символьных константах и комментариях.

Операторы

"Операторы" это символы (состоящие из одного символа или комбинации символов), которые задают манипуляции над значениями. Каждый символ интерпретируется как отдельный элемент, называемый "лексемой".

В Таблице 2 содержится список унарных операторов Вычислителя с их именами. В Таблице 3 содержится список бинарных операторов с их именами. Операторы нужно задавать так, как они показаны в таблицах, без разделительных знаков в многосимвольных операторах. Обратите внимание на то, что знак минуса появляется в двух таблицах. Его унарная или бинарная интерпретация зависит от контекста, в котором он появляется.

Таблица 2. "Унарные операторы"

Оператор	Название
!	логическое НЕ
-	арифметическое

	отрицание
++	арифметическое инкрементирование
--	арифметическое декрементирование

Таблица 3. "Бинарные операторы"

Оператор	Название	Оператор	Название
+	сложение	!=	неэквивалентность
-	вычитание	AND	логическое И
*	умножение	OR	логическое ИЛИ
/	деление	XOR	логическое исключающее ИЛИ (XOR)
&	битовое И	=	простое присвоение
|	битовое ИЛИ	+=	присвоение со сложением
^	битовое исключающее ИЛИ (XOR)	-=	присвоение с вычитанием
%	остаток от деления	*=	присвоение с умножением
<=	меньше или равно	/=	присвоение с делением
>	больше	&=	присвоение с битовым И
>=	больше или равно	|=	присвоение с битовым ИЛИ
==	эквивалентно (равно)	^=	присвоение с битовым XOR
$	Вхождение левого строкового оператора в правый. Проверка, входит ли подстрока в другую строку.		

Константы

Константа это число, символ или строка символов, которая в программе используется, как значение. Значение константы нельзя изменить.

В Вычислителе есть четыре вида констант: целые, с плавающей точкой, символьные и константы с типом дата и время.

Целые константы

Синтаксис: *цифры*
0xцифры
0Xцифры

"Целая константа" это десятичное, восьмеричное или шестнадцатеричное число, которое представляет собой целое значение в одной из следующих форм:

- "Десятичная константа" имеет форму одной или нескольких цифр (от 0 до 9).
- "Шестнадцатеричная константа" имеет форму 0xцифры или 0Xцифры, где цифры это одна или несколько шестнадцатеричных цифр (от 0 до 9 и прописные или строчные буквы от a до f), указание 0x или 0X обязательно.

Таблица 4. Примеры целых констант

Десятичные	Шестнадцатеричные
10	0xa или 0xA

132	0x84
32179	0x7db3 или 0x7DB3

Целые константы всегда имеют положительные значения. Если нужно использовать отрицательное значение, то поместите знак минус (-) перед константой для формирования выражения с отрицательным значением. В данном случае знак минус интерпретируется как унарный арифметический оператор отрицания.

Каждой целой константе на основании ее значения присваивается соответствующий тип. Целые константы представленные в десятичном виде всегда являются целым числом с диапазоном значений от -2147483648 до 2147483647. Тип целой константы представленной в шестнадцатеричном виде определяется ее значением. Если значение не превосходит 0xFFFFFFFF, то считается что константа является целым числом. В противном случае, константа преобразуется в двоичный массив.

Константы с плавающей точкой

Синтаксис: *[цифры][.цифры][E|e-[+]цифры]*

"Константа с плавающей точкой" это десятичное число, которое соответствует действительному числу со знаком. Значение действительного числа со знаком состоит из целой части, дробной части и показателя степени. "Цифр" может не быть или их может быть несколько (от 0 до 9), а E (или e) это символ экспоненты. Можно опустить либо цифры до десятичной точки (целая часть числа) либо после десятичной точки (дробная часть числа), но не одновременно. Если используется показатель степени, то только в этом случае можно не вводить десятичную точку. Показатель степени состоит из символа экспоненты (E или e) за которым следует постоянное целое значение. Целое значение может быть отрицательным. Нельзя использовать разделительные символы между цифрами или символами константы.

Константы с плавающей точкой всегда имеют положительные значения. Если нужно использовать отрицательное значение, то поместите знак минус (-) перед константой для формирования выражения с отрицательным значением. В данном случае знак минус интерпретируется как арифметический оператор.

Все константы с плавающей точкой имеют тип вещественное число.

Следующие примеры иллюстрируют некоторые формы констант и выражений с плавающей точкой:

15.75
1.575E1
1575e-2
-0.0025
-2.5e-3
25E-4

Строковые константы

Синтаксис: *"символы"["символы"]...*

Строковая константа - это последовательность символов, которая заключена в двойные или одинарные кавычки. Например, строковая константа может быть такой:

"Это строковая константа."

В строковой константе "символы" могут содержать любые символы, в том числе перевод строки / возврат каретки. При необходимости указать внутри константы кавычку, которая была использована в качестве ограничителя константы, необходимо дважды повторить данный символ. Например.

"ООО ""Ромашка"" "

Традиционный способ формирования строк литералов, которые не умещаются на одной строке, состоит в том, что константа разбивается на две части и между ними добавляется знак сложения (+). Например, строковая константа

"Эта длинная константа "+
"не помещается на одной строке."

идентична строке

"Эта длинная константа не помещается на одной строке."

Константы, имеющие тип дата и время

Синтаксис: *{ДД.ММ.ГГГГ ЧЧ:ММ:СС}*

Константа, имеющая тип дата и время – это дата и время, записанное десятичными цифрами с использованием в качестве разделителя дней, месяцев и годов точку (.) или знак деления (/), а в качестве разделителя часов, минут и секунд двоеточие (:). При этом дата должна быть отделена от времени пробелом. Поддерживается три формата представления даты:

- ДД.ММ.ГГ – два символа номера дня, два символа номера месяца и два символа номера года
- ДД.ММ.ГГГГ – два символа номера дня, два символа номера месяца и четыре символа номера года
- ГГГГ.ММ.ДД – четыре символа номера года, два символа номера месяца и два символа номера дня

Время может быть не указано. Если время указано, то оно обязательно должно содержать часы и минуты. Секунды могут быть пропущены. Константы данного типа должны быть расположены в одной строке. Перенос на другую строку не допускается. Между символом открывающей фигурной скобки и первой цифрой могут располагаться только символы пробела. Другие символы разделители не допускаются. Примеры констант с типом дата и время:

- {01.01.2010 12:00:00}
- {2010-03-31 11:58}
- {2010-03-31}

Константы, задаваемые в справочнике констант системы

Синтаксис: *_ИмяКонстанты*

Система Айтида в своем составе имеет справочник констант. В данном справочнике пользователь имеет возможность описать необходимые ему константы без использования программирования скриптов. Это удобно, когда необходимо определить какие-либо параметры, используемые во всей

системе. Например. Значение константы ***_БУХУЧЕТ*** определяет, ведется ли в системе Айтида бухгалтерский учет. Значение этой константы может использоваться в любых скриптах и выражениях используемых при настройке системы Айтида.

Отличительной особенностью использования таких констант, является то, что их значение может изменяться во времени, при этом для каждого интервала времени может быть задано свое значение константы. Для определения какой именно интервал времени для констант должен использоваться задается встроенной функцией **ДАТАКОНСТАНТ([Дата]) (CONSTANTSDATE([Дата]))**

Идентификаторы

Синтаксис: *буква|_|@[буква|цифра|_]...*

"Идентификаторы" - это имена, которые Вы присваиваете переменным и функциям в вашей программе. Идентификатор это последовательность из одной или нескольких букв, цифр, знаков подчеркивания (_) или символа @, которая обязательно начинается с буквы, подчеркивания или символа @. Идентификатор может содержать любое число символов. Идентификатор создается его заданием в объявлении переменной ключевым словом PUBLIC или LOCAL, или при первом присвоении ему значения. После этого его можно использовать в операторах программы для ссылки на связанный с ним элемент. Интерпретатор Вычислителя рассматривает прописные и строчные буквы как одинаковые символы, поэтому нельзя создавать разные идентификаторы с одинаковым произношением, но с одной или несколькими буквами иного размера. Идентификаторы не могут иметь произношение и написание, совпадающее с ключевыми словами языка. Ключевые слова описаны в разделе "Ключевые слова".

Понятия

Синтаксис: *@буква|_|@[буква|цифра|_]...*

Система Айтида в своем составе имеет справочник понятий. В данном справочнике пользователь имеет возможность описать необходимые ему понятия. Понятие – это поименованные правила получения информации из базы данных. Другими словами, понятия содержат в себе информацию о том, как получить значение того или иного элемента из базы данных. Например. В системе есть справочник товаров. Во многих печатных формах используется элемент «Наименование товара». Доступ к данному элементу всегда осуществляется через внутренний код товара в базу данных. Для того чтобы, каждый раз, когда требуется получить наименование товара не писать запрос к базе данных для получения этого наименования, можно описать понятие @НаименованиеТовара, в котором указать правило (запрос) для получения наименования, используя код товара. После этого везде, где необходимо, можно будет использовать вместо запроса к базе данных созданное понятие. Понятия могут использовать в своем описании другие понятия. Уровень вложенности таких описаний не ограничен. Также для понятия можно определить значение по умолчанию, которое будет использовано в случае, если для вычисления значения понятия не хватает какой-либо информации.

Понятия могут быть трех типов.

- Базовое понятие. Это понятие, которое заменяет собой имя в таблице базы данных. Например, понятие ***@Товар*** является ссылкой на поле ***пп***. Таким образом, когда Вычислитель встречает данное понятие, он ищет в текущем контексте вычислений поле или переменную с именем ***пп***. Если такая переменная или поле доступны, то будет использовано их значение. Если такого поля нет, то будет использовано значение, указанное в качестве значения по умолчанию для понятия.

- Понятие SQL выражение. Это понятие, для вычисления значения которого необходимо выполнить указанный в описании SQL запрос к базе данных. Например, понятие ***@НаименованиеТовара*** является SQL запросом к базе данных ***"SELECT name FROM sprnn WHERE nn= @Товар"***. Когда Вычислитель встречает данное понятие, он в первую очередь вычисляет значение использованного вложенного понятия ***@Товар***. Затем Вычислитель выполняет запрос к базе данных, используя текущее соединение, для определения названия товара.
- Понятие выражение. Это понятие, которое является выражением, значение которого необходимо вычислить. Например, понятие ***@ПолноеНаименованиеТовара*** можно определить как выражение ***@НаименованиеТовара + @ТоварОбъем***. Далее, в отчетных формах можно будет использовать созданное понятие. В случае, если в будущем возникнет необходимость выводить в отчетах не объем товара, а например, его вес, то для этого не надо будет менять большое количество отчетных форм, а достаточно будет изменить правило формирования полного наименования товара.

Ключевые слова

"Ключевые слова" это заранее определенные идентификаторы, которые имеют для интерпретатора Вычислителя специальное значение. Их можно использовать только так, как они определены. Имя элемента программы не может совпадать по произношению и написанию с ключевым словом. В языке Вычислителя имеются следующие ключевые слова:

IF, WHILE, BREAK, CONTINUE, TRY, CATCH, THROW, BEGIN, END, RETURN, PUBLIC, LOCAL, FUNCTION, ФУНКЦИЯ, CONCEPT, ПОНЯТИЕ, CONSTANT, КОНСТАНТА, OBJECT, ОБЪЕКТ, SQL, SQLAGGREGATE, SQLERROR

Блок команд.

Блок команд может состоять из одной или нескольких команд, разделенных точкой с запятой ; Весь блок заключается в фигурные скобки. Если в блоке только одна команда, то её можно не заключать в фигурные скобки. Если блок команд заключен в фигурные скобки, то после закрывающей фигурной скобки точка с запятой не требуется. А перед ней требуется.

Блок команд может не содержать команд. В этом случае требуется заключать его в фигурные скобки.

Примеры.

```
if ( Условие )
        {}              // Блок команд без самих команд.
if ( Условие )
        return false;   // Блок команд из одной команды без фигурных скобок.
if ( Условие )
{
        // Блок команд из нескольких команд, заключенных в фигурные скобки.

        Сообщение( "Условие не выполнено." );
        return false;
}
```

Ключевые слова.

Слово IF

Условный оператор, который позволяет выполнять действия только в том случае, если выполняется указанное в нем Условие.

Синтаксис

IF (Условие)
    БлокКомандИстина
[ELSE БлокКомандЛожь]

Условный оператор.

- Условие. Выражение, значение которого может быть интерпретировано как Истина или Ложь. Допустимые типы
    - Логический
    - Числовой. Значение 0 будет интерпретировано как Ложь. Все остальные – как Истина.
    - Другие типы значений вызовут ошибку исполнения программы.
- БлокКомандИстина. Блок команд, которые будут выполнены в случае, если значение Условия будет интерпретировано как Истина. Этот блок команд должен присутствовать обязательно.
- Ключевое слово ELSE. Если требуется выполнить блок команд в случае невыполнения условия, то необходимо добавить это ключевое слово. Если такой необходимости нет, то ключевое слово ELSE можно не указывать.
- БлокКомандЛожь. Блок команд, которые будут выполнены в случае, если значение Условия будет интерпретировано как Ложь. Этот блок команд должен присутствовать обязательно, если указано ключевое слово ELSE. Иначе, его не должно быть.

Слово WHILE (ПОКА)

Условный оператор цикла, который позволяет выполнять действия до тех пор, пока выполняется указанное в нем УсловиеЦикла.

Синтаксис

WHILE (УсловиеЦикла)
    БлокКоманд

Либо
ПОКА (УсловиеЦикла)
    БлокКоманд

Условный оператор цикла.

- УсловиеЦикла. Выражение, значение которого может быть интерпретировано как Истина или Ложь. Допустимые типы
    - Логический
    - Числовой. Значение 0 будет интерпретировано как Ложь. Все остальные – как Истина.
    - Другие типы значений вызовут ошибку исполнения программы.

  Значение выражения будет вычисляться при входе в цикл и каждый раз при завершении выполнения блока команд. БлокКоманд должен либо влиять на значение выражения, либо,

содержать ключевое слово BREAK, для принудительного выхода из цикла. Иначе, цикл будет бесконечным.

- БлокКоманд. Блок команд, которые будут выполнены в случае, если значение УсловияЦикла выполнилось. Этот блок команд должен присутствовать обязательно.

Слово *BREAK*

Принудительный выход из цикла. Если блок команд цикла содержит ключевое слово BREAK, то при его выполнении будет произведен выход из цикла, без выполнения оставшихся команд блока.

Синтаксис

```
WHILE ( УсловиеЦикла )
{
	Команды блока;
	BREAK;
	Команды блока;
}
```

Принудительный выход из цикла. Обычно это слово помещают в условный оператор, чтобы при выполнении некоего условия прервать выполнение цикла.
Пример.

```
	WHILE ( Условие )
	{
		IF ( СООБЩЕНИЕ( "Продолжить?", "Ожидание", 4 ) == 6 ) // Если нажали "Да"
		{
			// повторная попытка соединения
		}
		ELSE  // Если нажали "Нет"
		{
			BREAK;
		}
	}
```

В данном примере в цикле производятся попытки соединения. Также у пользователя запрашивается согласие на продолжение ожидания соединения. Если пользователь не согласен, то производится принудительный выход из цикла.

Слово *CONTINUE*

Принудительный повтор цикла, без выполнения последующих команд блока. Если блок команд цикла содержит ключевое слово CONTINUE, то при его выполнении будет произведен возврат к проверке условия цикла, без выполнения оставшихся команд блока.

Синтаксис

```
WHILE ( УсловиеЦикла )
{
	Команды блока;
	CONTINUE;
	Команды блока;
}
```

Принудительный повтор цикла. Обычно это слово помещают в условный оператор, чтобы при выполнении некоего условия не выполнять другие команды блока.
Пример.

```
// Не чеки продажи/возврата/покупки/возврата покупки пропускаем
WHILE ( !КОНЕЦКОНТЕКСТА( "Чеки" ) )
{
	IF ( !ВСПИСКЕ( Чеки.checktype, 0, 1, 2, 3, 6, 11, 12 ) )
	{
		//------- Если нашли чек,…
		ПРОПУСТИТЬ( 1, "Чеки" );
		CONTINUE;
	}
	m_doccode		= Чеки.doccode;
	m_docic			= 0;
	// Дальнейшая обработка
}
```

В данном примере в цикле производится обработка загруженных чеков. Но, некоторые типы чеков не нуждаются в такой обработке, и они пропускаются. Команды, предшествующие ключевому слову CONTINUE должны как-то влиять на условие цикла, для того чтобы не произошло зацикливание. В данном примере, роль изменения условия играет переход на следующую запись просматриваемой таблицы.

Слово *RETURN*

Принудительный выход из функции. Если блок команд функции содержит ключевое слово RETURN, то при его выполнении будет произведен выход из функции, без выполнения оставшихся команд блока.

Синтаксис

RETURN [Выражение];

Принудительный выход из функции. Если указано Выражение, то перед возвратом из функции оно будет вычислено и его результат будет считаться результатом выполнения функции.
Пример.

```
IF ( ПУСТО( Токен ) )
{
	СООБЩЕНИЕ( "Не определен токен.", "Отправка чека");
	RETURN false;
}
```

В данном примере в случае отсутствия обязательного параметра выводится сообщение и выполняется выход из функции со значением Ложь.

Слово *TRY/CATCH*

Оператор TRY/CATCH позволяет организовать безопасное выполнение блока команд. Безопасное – означает, что в случае возникновения ошибок выполнения в блоке команд, работа всей программы не завершится ошибкой, а будет продолжена либо с команд блока CATCH, либо с команд, следующих за блоком TRY. К ошибкам исполнения относятся все возможные проблемы, которые могут

произойти. Это могут быть ошибки вычисления выражений. Например, в выражении может быть указана неизвестная переменная, тогда это вызовет исключение. Либо, вычисление выражения привело к недопустимому индексу в массиве. Либо возникли проблемы во внешних функциях, и они вызвали общее исключение защиты. Во всех этих случаях, выполнение программы не будет прервано, если блок команд заключен в TRY. Также возможна программная генерация исключения в необходимых случаях.

Синтаксис

TRY
	БлокКоманд
[
CATCH [(Переменная)]
	БлокКоманд
]

Пример.

```
try
{
    ТекстОшибки         = "Сервер " + АдресСервера + CHR( 13 ) + "вернул сообщение об ошибке: " + CHR( 13 );
    Соединение          = HTTPCONNECT( АдресСервера, "", true, BonusAPI.ИМЯФАЙЛАЖУРНАЛА( "", "TEYCA_api_log", true ) );
    Ответ               = ПЕРЕКОДИРОВАТЬ( HTTPPOST( Соединение, Ресурс, ТелоЗапроса, "Заголовки" ), "UTF-8", "ANSI" );
    //сообщение( ответ );
    IF ( !BonusAPI.TEYCA_ПРОВЕРКАНАОШИБКИ( @Ответ ) ) THROW ( Ответ );
}
catch ( ТекстСообщения )
{
    HTTPCLOSE( Соединение );
    IF ( !ПУСТО( ТекстСообщения ) )
        СООБЩЕНИЕ( ТекстСообщения, "TEYCA" );
    RETURN "";
}
HTTPCLOSE( Соединение );
RETURN Ответ;
```

В данном примере в защищаемый блок помещены команды соединения с сервером по протоколу HTTP. Исключение может возникнуть как во внешнем контуре, связанном с HTTP клиентом, так и в результате получения ошибочного ответа от сервера. Во втором случае вызывается программное исключение словом THROW. Команды блока CATCH отработают в случае любого из возникших исключений. Если это было программное исключение, т.е. был передан текст сообщения об ошибке, то будет выведено сообщение об ошибке. Иначе, будет выведено общее сообщение об ошибке.

Слово *PUBLIC*

Объявление глобальной переменной. При обычном объявлении переменной (присваивании ей значения) переменная становится локальной и уничтожается при завершении выполнения функции. В случае, если переменную объявить глобальной, то она будет доступна во всех функциях и не будет уничтожаться ни при каких условиях.

Синтаксис

PUBLIC ИмяПеременной [, ИмяПеременной[,…]];

Объявление глобальной переменной. Список имен переменных может содержать несколько переменных. По умолчанию переменные объявляются логическими со значением Ложь. Переменная

может быть объявлена глобальной после присвоения ей значения. В таком случае, она будет переведена из локальной в глобальную область.

Пример.
PUBLIC ИмяФайлаЖурнала, АдресСервера;

```
PUBLIC ИНДЕКСУСТРОЙСТВАККМ, _ИМЯКАССИРА, _ТИПДОКУМЕНТАОСНОВАНИЯ, _ИДДОКУМЕНТАОСНОВАНИЯ, _ПЕЧАТАТЬНАЛОГИ, _ПЕЧАТАТЬСКИДКИ, _ОПРЕДЕЛЯТЬКОДЫНАЛОГОВ, ФискальныйЧек
PUBLIC ВСЕГООБЕЗНАЛ, ВСЕГОНАЛ, ВСЕГОКРЕДИТ, ВСЕГОСЕРТИФИКАТЫ, ВСЕГОПРОЧАЯОПЛАТА;
PUBLIC _СТАВКАНДС_БЕЗНДС, НомерЧекаККМ, НомерСменыККМ, КОЛИЧЕСТВОСИМВОЛОВЧЕКОВОЙЛЕНТЫ, ШРИФТТЕКСТАДЛЯНЕФИСКАЛЬНОЙПЕЧАТИ;
PUBLIC _РАБОТАСЕГАИС, _ККТ_ШИРИНА_ЛЕНТЫ, _СУММАСКИДКИВСЕГО;
PUBLIC ИмяТаблицыСтрокЧека, _ПЕЧАТАТЬСКИДКИ, ТекстЧека, _ВСЕГООПЛАТАБОНУСАМИ, _ВСЕГОСУММАЧЕКАРАСЧЕТНАЯ, ВерсияФФД;
PUBLIC _КОЛИЧЕСТВОКОПИЙСЛИПА, НомерЛУПлатежнойСистемы, TerminalNumber, ЕстьНеявныеОшибкиПС, ВанковскиеПозицииЧека, ИДТЕРМИНАЛАИЗПРОФИЛЯ, _ККТСУММАБЕЗНАЛ;
```

В данном примере объявляются переменные, значения которых будут доступны во всех функциях и скриптах.

Слово *__SQL*

Слово позволяет писать текст SQL запросов не в константных строках, а как простой текст запроса с подсветкой синтаксиса.

Синтаксис.

```
__SQL( [ИмяКонтекста] )
{
        ТекстЗапроса
}
[
SQLERROR
        БлокКоманд
]
```

В заголовке ключевого слова __SQL может быть указано имя контекста, в который необходимо поместить результат выполнения запроса. Текст запроса может содержать вхождения вида [[Выражение]]. Значения выражения будут вычислены, приведены к строковым выражениям и подставлены в запрос. В случае возникновения ошибки выполнения запроса будет выполнен блок команд из слова SQLERROR.

```
IF ( !ПУСТО( НовоеСвойствоЧека ) )
{
    try
    {
        __SQL( )
        {
            UPDATE chequelist
            SET pex = '[[ STDF( НовоеСвойствоЧека ) ]]'
            WHERE fr_code = '[[ КОДОБОРУДОВАНИЯ ]]'
                AND identity_column = [[ ИДЕНТИФИКАТОРЧЕКА ]]
        }
        SQLERROR
        {
            СООБЩЕНИЕ( _ERRORDESCRIPTION, "ошибка записи фискальных данных в чек" );
            RETURN false;
        }
    }
    catch( )
    {
        РезультатЗаписиСвойств  = false;
    }
}
```

```
__SQL( "mark_in_spec" )
{
    -- ТАБЛИЦА  С ТОВАРАМИ ДЛЯ ПОСТУПЛЕНИЯ КОДОВ МАРКИРОВКИ СИСТЕМЫ МЕРКИРОВКА
    INSERT INTO mark_in_spec ( ic, ent, linenum, nn, name, gtin, cena, kolp, summa, ed, okei, kodn, barcode, article, country )
    VALUES ( '[[ ИДДокумента ]]', '[[ STDF( Предприятие ) ]]', [[ НомерСтроки ]], '[[ Товар ]]', '[[ STDF( НаименованиеТовара, 250 ) ]]',
             '[[ STDF( мGTIN, 15 ) ]]', [[ STR( Цена, 16, 6 ) ]], [[ STR( Количество, 16, 3 ) ]], [[ STR( Сумма, 16, 3 ) ]],
             '[[ STDF( КодЕдиницы, 15 ) ]]', '[[ STDF( ОКЕИ, 15 ) ]]', '[[ STDF( КодНалога ) ]]', '[[ STDF( мШтрихКод ) ]]',
             '[[ STDF( Артикул, 50 ) ]]', '[[ STDF( СтранаПроисхождения, 10 ) ]]' );
}
SQLERROR throw( 1 );
```

Комментарии

Синтаксис: *`/* символы */`*

"Комментарий" это последовательность символов, которая рассматривается интерпретатором как разделительный символ, но игнорируется им. "Символы" в комментарии могут включать в себя любые комбинации символов, включая символ новой строки, кроме ограничителя "конец комментария" (*/). Комментарии могут занимать более одной строки, но не могут быть вложенными. Комментарии могут появляться везде, где допустимо появление разделительных символов. Комментарий рассматривается интерпретатором как единичный разделительный символ, поэтому нельзя включать комментарии с лексемами. Однако, из-за того, что компилятор игнорирует символы комментария, в его текст можно включать ключевые слова без появления ошибок.

Приведем ряд примеров, иллюстрирующих комментарии:

```
/* Комментарии могут располагаться на
  Нескольких строках. */
/* Комментарии могут содержать любые символы
  В том числе ключевые слова, например while. */
```

Комментарии не могут содержать вложенные комментарии. Следующий пример вызовет появление ошибки:

```
/* Нельзя создавать /* вложенные */ комментарии */
```

Ошибка происходит из-за того, что интерпретатор распознает первое появление */ после слова nest как конец комментария. Он пытается обработать остальной текст и не может этого сделать, поэтому выдает сообщение об ошибке.

Лексемы

В исходном тексте программы интерпретатор распознает основные элементы из групп символов, называемые "лексемами". Лексема — это текст исходной программы, который интерпретатор не анализирует на входящие в него компоненты. Например, в следующем фрагменте программы слово elsewhere использовано в качестве имени функции. Хотя else и является ключевым словом, конфликта между лексемой имени функции и лексемой ключевого слова не происходит.

```
ФУНКЦИЯ main( )
{
        i = 0;
        if ( i )
                elsewhere( );
}
```

Однако, если ввести elsewhere как else where, с пробелом между словами else и where, то в предыдущем примере Вычислитель будет пытаться выполнить функцию where( ), т.к. указанное условие не выполнится. Примерами лексем могут служить описанные в данной главе операторы, константы, идентификаторы и ключевые слова. Также являются лексемами такие символы пунктуации, как прямоугольные ([]), фигурные ({}) и угловые скобки (<>), двоеточие и запятая. Лексемы отделяются разделительными символами и другими лексемами, например, операторами и символами пунктуации. Для предотвращения разбивки интерпретатором элементов на две или несколько лексем, в идентификаторах, многосимвольных операторах и ключевых словах запрещено использование разделительных символов. Когда интерпретатор интерпретирует лексемы, он пытается включить в отдельную лексему как можно больше символов до перехода к следующей лексеме. Из-за этой особенности интерпретатор может неверно интерпретировать лексемы, если они не выделяются разделительными символами. Пример. Рассмотрим следующее выражение:

```
i+++j
```

В данном примере интерпретатор сначала пытается выделить максимально длинный оператор (++) из двух знаков плюс. Оставшийся оператор плюс рассматривается как оператор сложения (+). Т.о. выражение интерпретируется как (i++)+(j), а не как (i)+(++j). В данном случае и в аналогичных случаях нужно использовать разделительные символы и скобки для исключения неоднозначности и обеспечения корректного вычисления выражений.

ФУНКЦИИ

Введение

Функция это независимый набор объявлений и операторов, который обычно разрабатывается для выполнения конкретной задачи. В определении функции задается ее имя и число ее формальных параметров и приводятся объявления и операторы, которые задают ее действие. Эти объявления и операторы называются "телом функции". Определение функции может не содержать ее тела, а указывать на то, что функция находится в DLL и должна быть загружена из указанного файла в момент обращения к ней. Объявления формальных параметров задают имена для формальных параметров. Сфера действия этих имен заканчивается в конце функции. При вызове функции управление выполнением программы передается на вызванную функцию. Действительные аргументы, если они есть, передают свои значения вызванной функции. Выполнение оператора return в вызванной функции возвращает управление и возможное возвращаемое значение на вызов функции.

```
    /** Объявление и определение функции **/
ФУНКЦИЯ СложениеСтрок ( a, b )
{
        IF ( ТИП( "a" ) != "C" OR  ТИП( "a" ) != "C" ) RETURN "";
        RETURN a + b;
}
```

Определения функций

Синтаксис:

```
FUNCTION [LOCAL | ЛОКАЛЬНАЯ] ИмяФункции([ИмяПараметра1[, ИмяПараметра2[,...]]])
{
      блок команд;
}
```

В "определении функции" задается ее имя, формальные параметры и тело функции. "Список-формальных-параметров" это последовательность имен формальных параметров, разделенных запятыми. Объявление формального параметра не содержит типа принимаемого значения. Поэтому, функция может быть вызвана с параметрами любого типа. Количество фактически переданных параметров может быть меньше количества формальных параметров. В этом случае недостающие параметры получат пустое значение при вызове функции. Если число фактических параметров больше чем формальных, то вызов функции не будет осуществлен и интерпретатор выдаст сообщение об ошибке.

Вызовы функций

Синтаксис: *выражение([список-выражений])*

"Вызов функции" это выражение, которое передает управление и действительные аргументы, если они есть, функции. В вызове функции "выражение" вычисляется для получения адреса функции, а "список-выражений" это список разделенных запятыми выражений. Значения этих выражений являются действительными аргументами, передаваемыми в функцию. Если у функции нет аргументов, то список выражений может быть пустым. При выполнении вызова функции:

1. Вычисляются выражения из списка выражений и преобразуются по правилам обычных арифметических преобразований.

2. Выражения из списка выражений передаются в формальные параметры вызванной функции. Первое выражение списка всегда соответствует первому формальному параметру функции, второе выражение соответствует второму формальному параметру, и т.д. по списку. Вызванная функция использует копии действительных аргументов, поэтому все изменения, которые она сделает с аргументами, не повлияют на значения переменных, с которых были сделаны копии.
3. Управление выполнением передается на первый оператор функции.
4. Выполнение оператора **RETURN** в теле функции возвращает управление и возвращаемое значение, если оно есть, на вызов функции. Если оператор **RETURN** не выполняется, то управление передается на вызов после выполнения последнего оператора вызванной функции. В этом случае возвращаемое значение является логическим и равно **ИСТИНЕ**.

Примечание. Выражения в списке аргументов функции могут быть вычислены в любом порядке, поэтому аргументы, значения которых могут быть изменены побочными эффектами из других аргументов, имеют неопределенные значения. Точка упорядочивания, задаваемая оператором вызова функции, гарантирует только то, что все побочные эффекты в списке аргументов вычисляются до передачи управления на вызванную функцию. Единственное требование к вызову функции состоит в том, что выражение до скобок должно вычисляться в адрес функции. Функция вызывается практически аналогично своему объявлению. Например, при объявлении функции задается имя функции, за которым в скобках следует список формальных параметров. Аналогично, при вызове функции нужно задать ее имя, за которым следует список аргументов в скобках.

Действительными аргументами могут быть любые значения. Все действительные аргументы передаются их значениями. Копия действительного аргумента передается соответствующему формальному параметру. Функция использует эту копию без какого-либо воздействия на переменную, из которой она была получена.

ПРИМЕРЫ СКРИПТОВ

Текст обработки загрузки курсов валют

```
// Загрузка информации о курсах валют производится с сайта ЦБ РФ
// Информация поступает в виде XML файла, в котором указана дата и курс на эту дату.
// В запросе информации необходимо указать период и код валюты
// Коды валюты - R01235 для USD, R01239 для EUR
// Будем запрашивать курсы за последние 7 дней. Сначала для USD, а затем для EUR

Соединение      = СокетСоединить( "www.cbr.ru", 80, true );
СокетПередать( Соединение, "GET /scripts/XML_dynamic.asp?date_req1=" + DTOC( ДОБАВИТЬДНИ( ДАТАВРЕМЯ(
), - 7 ), 4 ) + "&date_req2=" +  DTOC( ДАТАВРЕМЯ( ), 4 ) + "&VAL_NM_RQ=R01235 HTTP/1.1" + CHR( 13 )
+ CHR( 10 ) + "Host: www.cbr.ru" + CHR( 13 ) + CHR( 10 ) + CHR( 13 ) + CHR( 10 ) );
Ответ           = СокетПолучить( Соединение, CHR( 13 ) + CHR( 10 ) + CHR( 13 ) + CHR( 10 ));

IF ( AT( Ответ, 'Content-Length:' ) > 0 )
{
        Длина   = VAL( SUBSTR( Ответ, AT( Ответ, 'Content-Length:' ) + 15 ));
        Ответ   = СокетПолучить( Соединение, Длина);
}
ELSE
{
        КОДРЕЗУЛЬТАТА           = false;
        ОПИСАНИЕРЕЗУЛЬТАТА      = "Ошибка чтения файла с сайта ЦБ РФ. " + _ERRORDESCRIPTION;
        RETURN false;
}

СокетЗакрыть( Соединение );
ИмяФайла        = УникальноеИмя( );
```

```
Файл            = ФайлСоздать( ИмяФайла );
ФайлЗаписать( Файл, Ответ );
ФайлЗакрыть( Файл );
ДобавитьКонтекст( "SET FMTONLY ON
                   SELECT GETDATE() AS date, CONVERT( float, 0 ) AS nominal,
                          CONVERT( float, 0 ) AS value SET FMTONLY OFF", "Курсы" );
Загрузить( "Курсы", "XML", ИмяФайла, "Record" );
// Если в справочнике валют указано, что используется обратный курс,
// то надо записывать обратный курс
ОбратныйКурс    = ЗАПРОС( "SELECT f_rev FROM sprcur WHERE code = 'USD'" );
ВыбратьКонтекст( "Курсы" );
ПерейтиВНачало( "Курсы" );
WHILE ( !КонецКонтекста( "Курсы" ) )
{
        IF ( Курсы.Value <> 0 )
        {
                ЗАПРОС( "DELETE FROM sprrate
                         WHERE DATEDIFF( day, date, '" + Курсы.date + "' ) = 0 AND code = 'USD'" );
                ЗАПРОС( "INSERT INTO sprrate ( code, date, rate )
                         VALUES ( 'USD', '" + Курсы.date + "', " +
                         STR( ЕСЛИ( ОбратныйКурс, 1 / Курсы.Value, Курсы.Value ), 20, 8 ) + ")" );
        }
        SKIP( );
}
УдалитьКонтекст( "Курсы" );
ФайлУдалить( ИмяФайла );

// Полностью аналогично для евро
Соединение      = СокетСоединить( "www.cbr.ru", 80, true );
СокетПередать( Соединение, "GET /scripts/XML_dynamic.asp?date_req1=" + DTOC( ДОБАВИТЬДНИ( ДАТАВРЕМЯ(
), - 7 ), 4 ) + "&date_req2=" + DTOC( ДАТАВРЕМЯ( ), 4 ) + "&VAL_NM_RQ=R01239 HTTP/1.1" + CHR( 13 ) +
CHR( 10 ) + "Host: www.cbr.ru" + CHR( 13 ) + CHR( 10 ) + CHR( 13 ) + CHR( 10 ) );
Ответ           = СокетПолучить( Соединение, CHR( 13 ) + CHR( 10 ) + CHR( 13 ) + CHR( 10 ));

IF ( AT( Ответ, 'Content-Length:' ) > 0 )
{
        Длина   = VAL( SUBSTR( Ответ, AT( Ответ, 'Content-Length:' ) + 15 ));
        Ответ   = СокетПолучить( Соединение, Длина);
}
ELSE
{
        КОДРЕЗУЛЬТАТА           = false;
        ОПИСАНИЕРЕЗУЛЬТАТА      = "Ошибка чтения файла с сайта ЦБ РФ. " + _ERRORDESCRIPTION;
        RETURN false;
}

СокетЗакрыть( Соединение );
ИмяФайла        = УникальноеИмя( );
Файл            = ФайлСоздать( ИмяФайла );
ФайлЗаписать( Файл, Ответ );
ФайлЗакрыть( Файл );
ДобавитьКонтекст( "SET FMTONLY ON
                   SELECT GETDATE() AS date, CONVERT( float, 0 ) AS nominal,
                          CONVERT( float, 0 ) AS value SET FMTONLY OFF", "Курсы" );
Загрузить( "Курсы", "XML", ИмяФайла, "Record" );
// Если в справочнике валют указано, что используется обратный курс,
// то надо записывать обратный курс
ОбратныйКурс    = ЗАПРОС( "SELECT f_rev FROM sprcur WHERE code = 'EUR'" );
ВыбратьКонтекст( "Курсы" );
ПерейтиВНачало( "Курсы" );
WHILE ( !КонецКонтекста( "Курсы" ) )
{
        IF ( Курсы.Value <> 0 )
        {
                ЗАПРОС( "DELETE FROM sprrate
```

```
                        WHERE DATEDIFF( day, date, '" + Курсы.date + "' ) = 0 AND code = 'EUR'" );
                ЗАПРОС( "INSERT INTO sprrate ( code, date, rate )
                        VALUES ( 'EUR', '" + Курсы.date + "', " +
                        STR( ЕСЛИ( ОбратныйКурс, 1 / Курсы.Value, Курсы.Value ), 20, 8 ) + ")" );
        }
        SKIP( );
}
УдалитьКонтекст( "Курсы" );
ФайлУдалить( ИмяФайла );
```

Скрипт автоматического создания документа Выпуск и комплектация при записи Документа кассовой смены

```
// Данный скрипт предназначен для использования в точке зрения
// в проверке документа кассовой смены перед записью.
// Загружаем реквизиты документа
ДобавитьКонтекст( "SELECT ndok, date, firm, rs, sklad, cur, cur_rate, branch, identity_column
                   FROM spr008
                   WHERE identity_column= " + @ИДДокумента, "Документ" );
ОсновнойУчет   = ЗАПРОС( "SELECT dbo.fn_getdefaultaccount ( '" + _ТОЧКАЗРЕНИЯ + "' )" );

// Если у документа есть выписанные на основании документы
// выпуска и комплектации, то ничего не делаем
IF ( ЗАПРОС( "SELECT CASE WHEN EXISTS(SELECT * FROM crosslink
                                      WHERE code = '008' AND ic= " + @ИДДокумента + " AND
                                            cr_code = '009') THEN 1 ELSE 0 END " ) == 1 )
        RETURN true;

ИмяТаблицыПродукции           = "#" + УникальноеИмя( );
//------- Создаем таблицу для сбора необходимой информации
ЗАПРОС( "CREATE TABLE " + ИмяТаблицыПродукции + "
               (nn char(10), nnname char(250), kolp float, incena float, cena float, ed char(15),
                kodn char(2), koef_e float, koef_c float, sklad char( 3 ) ) " );
СистемноеСообщение( "Пожалуйста, подождите. " +
                    "Идет проверка документа на наличие производимой продукции." );

ДобавитьКонтекст( "SELECT nn, nnname, SUM( kolp ) AS kolp, ed, kodn, koef_e, koef_c
                   FROM spec008
                   WHERE ic= " + @ИДДокумента + "
                   GROUP BY nn, nnname, ed, kodn, koef_e, koef_c ", "Спецификация" );

ВыбратьКонтекст( "Спецификация" );
ИтогоСумма             = 0.00;
WHILE (  !КонецКонтекста( "Спецификация" ) )
{
        //------- Если в карточке товара не стоит галочка Формировать автоматически - пропускаем
        IF ( ЗАПРОС( "SELECT f_make FROM sprnn WHERE nn= '" + Спецификация.nn + "' " ) )
        {
                //------- Проверяем наличие необходимого количества уже собранного на складе
                ДобавитьКонтекст( "EXECUTE sp_calcsklad '" + Спецификация.nn + "', '" +
                                  ТТОС( Документ.date ) + "', '" + Документ.sklad + "', '" +
                                  ОсновнойУчет + "'", "ОстатокНаДату" );
                ДобавитьКонтекст( "EXECUTE sp_calcsklad '" + Спецификация.nn + "', '', '" +
                                   Документ.sklad + "', '" + ОсновнойУчет + "'", "ТекущийОстаток" );

                // Если не хватает остатка, то добавляем строку в таблицу
                IF ( ROUND( Спецификация.kolp - ОстатокНаДату.kolp, 3 ) > 0 OR
                    ROUND( Спецификация.kolp - ТекущийОстаток.kolp, 3 ) > 0 )
                {
                        Количество     = MAX( ROUND( Спецификация.kolp - ОстатокНаДату.kolp, 3 ),
                                              ROUND( Спецификация.kolp - ТекущийОстаток.kolp, 3 ) );
                        Склад          = ЗАПРОС( "SELECT sklad_ingredients FROM sprres_g
                                                  WHERE code= (SELECT group_ FROM sprres
                                                  WHERE code= '" + Спецификация.nn + "')" );
```

```
                        ЗАПРОС( "INSERT INTO " + ИмяТаблицыПродукции + "
                                ( nn, nnname, kolp, cena, incena, ed, kodn, koef_e, koef_c, sklad )
                                VALUES ( '" + STDF( Спецификация.nn ) + "', '" +
                                STDF( Спецификация.nnname ) + "',           " +
                                STR( Количество, 20, 8 ) + ",               " +
                                STR( ОстатокНаДату.cena, 20, 8 ) + ",       " +
                                STR( ОстатокНаДату.cena, 20, 8 ) + ",       '" +
                                STDF( Спецификация.ed ) + "',               '" +
                                STDF( Спецификация.kodn ) + "',             " +
                                STR( Спецификация.koef_e, 20, 8 ) + ",      " +
                                STR( Спецификация.koef_c, 20, 8 ) + ",      '" + Склад + "' ) " );
                        ИтогоСумма     += ( Спецификация.kolp - ОстатокНаДату.kolp) *
                                          ОстатокНаДату.cena;
                }
                УдалитьКонтекст( "ОстатокНаДату" );
                УдалитьКонтекст( "ТекущийОстаток" );
        }
        Пропустить( 1, "Спецификация" );
}
УдалитьКонтекст( "Спецификация" );

//------ Начинаем транзакцию для записи документа Выпуск и комплектация
ЗАПРОС( "BEGIN TRANSACTION" );
//------ Создаем документ Выпуск и комплектация
//------ Код Выпуска и комплектации - 009
// Формируем список учетов для нового документа
СписокУчетов                                = "";
ДобавитьКонтекст( "SELECT account_ FROM specviewpoints_def
                   WHERE code= '" + _ТОЧКАЗРЕНИЯ + "' AND objtype= 'DOC' AND
                         objcode= '009' AND t_concept = ''", "МоделиУчета");
WHILE ( !КонецКонтекста( "МоделиУчета" ) )
{
        СписокУчетов                        += МоделиУчета.account_;
        Пропустить( 1, "МоделиУчета" );
}
УдалитьКонтекст( "МоделиУчета" );
СписокУчетов           = ALLTRIM( СписокУчетов );

ДобавитьКонтекст( "EXECUTE sp_insertdoc '009', '', '" + Документ.date + "', '', '" +
                        Документ.firm + "', '', 'RUB', '', '" + СписокУчетов + "', '" +
                        _ТОЧКАЗРЕНИЯ + "', 0, '" + ОсновнойУчет + "'", "ident" );

ДобавитьКонтекст(
        "UPDATE spr009 SET
                f_make  = 1,
                sklad   = '" + Документ.sklad + "',
                sklad_d = '" + Документ.sklad + "',
                summa   = "  + STR( ИтогоСумма, 20, 8 ) + ",
                firm    = '" + Документ.firm + "',
                rs        =  '" + Документ.rs + "',
                branch  = '" + Документ.branch + "',
                cur     = '" + Документ.cur + "',
                cur_rate= " + STR( Документ.cur_rate, 16, 8 ) + ",
                taccess = '', caccess='',
                note    = 'По документу кассовой смены  № " +
                                ALLTRIM( Документ.ndok ) + " от " + DTOC( Документ.date ) + "'
                WHERE identity_column= " + ident.ident, "Пустой" );
УдалитьКонтекст( "Пустой" );

ДобавитьКонтекст(
        "INSERT INTO spec009 ( ic, nn, spec_ic, ed, kolp, nnname, incena, cena, summa,
                               kodn, koef_e, koef_c, f_make, sklad )
         SELECT " + ident.ident + ", nn, 0, ed, kolp, nnname, incena, cena, kolp * cena,
                               kodn, koef_e, koef_c, 1, sklad
         FROM " + ИмяТаблицыПродукции, "Пустой" );
```

```
УдалитьКонтекст( "Пустой" );

//-------- Вызываем процедуру формирования состава продукции
ЗАПРОС("EXECUTE sp_recalc009 " + ident.ident );

// Добавляем связь на основании
ДобавитьКонтекст(
        "INSERT INTO crosslink (code, ic, ndok, date, cr_code, cr_ic, cr_ndok, cr_date, cr_type )
         VALUES ( '008'," + Документ.identity_column + ", '" + Документ.ndok + "', '" +
         TTOC( Документ.date ) + "', '009'," + ident.ident + ", '" + ident.ndok + "', '" +
         TTOC( ident.date ) + "', '0' )", "Пустой" );
УдалитьКонтекст( "Пустой" );

ЗАПРОС("IF @@TRANCOUNT > 0 COMMIT TRANSACTION");

СистемноеСообщение( );
УдалитьКонтекст( "Документ" );
//------ Вызываем созданный документ на редактирование
ОтправитьСообщение( _ГЛАВНОЕОКНОПРИЛОЖЕНИЯ, _СООБЩЕНИЕ_ПРОСМОТРДОКУМЕНТА, "009", ident.ident );

УдалитьКонтекст( "ident" );
// Возвращаем ИСТИНУ, т.к. необходимо произвести запись до конца
RETURN true;
```

Скрипт автоматического создания документа Выписка из банка, на основании отмеченных в журнале платежных поручений

```
// Скрипт создает документ Выписка из банка на основании
// платежных поручений, отмеченных в журнале документов
// Скрипт предназначен для использования в качестве дополнительной команды в пункте
// меню на правой кнопке мыши в журнале платежных поручений
СписокУчетов          = "001 002";
ДатаДокумента         = ДАТАВРЕМЯ( );
НомерДокумента        = "";
ФирмаДокумента        = "0000001";
АвторДокумента        = "";
ВалютаДокумента       = "RUB";
_ТОЧКАЗРЕНИЯ          = "0000000001";
КодДенежногоКармана   = "001";
РасчетныйСчет         = "12345678909876543210";

ИДБанковскойВыписки   = ЗАПРОС( "EXECUTE sp_insertdoc '012', '', '" + ДатаДокумента + "',
                                '" + НомерДокумента + "', '" + ФирмаДокумента + "',
                                '" + АвторДокумента + "', '" + ВалютаДокумента + "', '',
                                '" + СписокУчетов + "', '" + _ТОЧКАЗРЕНИЯ + "', 0, '001'", "ident" );

ЗАПРОС( "UPDATE spr012 SET
              kassakod      = '" + КодДенежногоКармана + "',
              firm          = '" + ФирмаДокумента + "',
              rs            = '" + РасчетныйСчет + "',
              summa         = (SELECT SUM( sumin - sumout )
                               FROM spec012 WHERE ic= " + ИДБанковскойВыписки + "),
              sumin         = (SELECT SUM( sumin - sumout )
                               FROM spec012 WHERE ic= " + ИДБанковскойВыписки + "),
              sumout        = (SELECT SUM( sumin - sumout )
                               FROM spec012 WHERE ic= " + ИДБанковскойВыписки + "),
              cur           = 'RUB',
              cur_rate      = 1,
              note          = 'Документ создан автоматически на основании '+
                              'отмеченных платежных поручений',
              author        = SUSER_SNAME( ),
              branch        = (SELECT branch FROM sprkskod
                               WHERE code = '" + КодДенежногоКармана + "' )
       WHERE identity_column = " + ИДБанковскойВыписки );
```

```
ДобавитьКонтекст( "INSERT INTO spec012 ( ic, client, contract, code_s, ic_s, ndok_s, date_s,
                                         note_s, sumout, branch )
                   SELECT " + ИДБанковскойВыписки + ", spr.client, spr.contract, spr.code,
                          spr.ic, spr.ndok, spr.date, spr.note, spr.summa, spr.branch
                   FROM spr000 spr
                   INNER JOIN " + _ИМЯТАБЛИЦЫ + " temp ON temp.code = spr.code AND
                                                          temp.ic = spr.ic", "СписокСтрок");
УдалитьКонтекст("СписокСтрок");

ОтправитьСообщение( _ГЛАВНОЕОКНОПРИЛОЖЕНИЯ, _СООБЩЕНИЕ_ПРОСМОТРДОКУМЕНТА, "012",
                    ИДБанковскойВыписки );
```

ВСТРОЕННЫЕ ФУНКЦИИ ВЫЧИСЛИТЕЛЯ

ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

СЖАТЬПРОБЕЛЫ(Строка)
ALLTRIM(Строка)
Убирает все лидирующие и завершающие пробелы из переданной строки.
Параметры.
Строка - Строка, из которой необходимо убрать пробелы.
Возвращаемое значение - Строка, полученная из исходной, без лидирующих и завершающих пробелов.

ПОЗИЦИЯ(СтрокаГдеИскать, СтрокаЧтоИскать[, НомерВхождения])
AT(СтрокаГдеИскать, СтрокаЧтоИскать[, НомерВхождения])
Возвращает номер позиции указанного (НомерВхождения) вхождения подстроки (СтрокаЧтоИскать) в исходную строку (СтрокаГдеИскать). Если вхождений не обнаружено, будет возвращен 0. Поиск осуществляется с учетом регистра.
Параметры.
СтрокаГдеИскать - Строка. В этой строке будет производиться поиск.
СтрокаЧтоИскать - Подстрока, которую необходимо найти в исходной строке.
НомерВхождения - Номер вхождения подстроки, которое необходимо найти. Если параметр не указан, то будет найдено первое вхождение.
Возвращаемое значение - число, номер позиции, начиная с 1, с которой начинается указанное вхождение подстроки в исходной строке. Если указанного вхождения не обнаружено, то функция вернет 0.

ПОЗИЦИЯР(СтрокаГдеИскать, СтрокаЧтоИскать[, НомерВхождения])
ATC(СтрокаГдеИскать, СтрокаЧтоИскать[, НомерВхождения])
Возвращает номер позиции указанного (НомерВхождения) вхождения подстроки (СтрокаЧтоИскать) в исходную строку (СтрокаГдеИскать). Если вхождений не обнаружено, будет возвращен 0. Поиск осуществляется без учета регистра.
Параметры.
СтрокаГдеИскать - Строка. В этой строке будет производиться поиск.
СтрокаЧтоИскать - Подстрока, которую необходимо найти в исходной строке.
НомерВхождения - Номер вхождения подстроки, которое необходимо найти. Если параметр не указан, то будет найдено первое вхождение.
Возвращаемое значение - число, номер позиции, начиная с 1, с которой начинается указанное вхождение подстроки в исходной строке. Если указанного вхождения не обнаружено, то функция вернет 0.

РАЗДЕЛИТЬ(Строка,Разделитель,ИмяМассива[,Флаг])
SPLIT(Строка,Разделитель,ИмяМассива[,Флаг])
Функция разделяет строку на подстроки, формируя массив с указанным именем. Возвращает количество сформированных элементов.
Параметры.
Строка - Строка, которую нужно разделить.
Разделитель - Строка, символы которой являются разделителями в первой строке.
ИмяМассива - Строка, имя формируемого массива.
Флаг - Число 0 или 1. Если 0, то пустые части не будут включаться в результат. Иначе - будут. По умолчанию - 0.
Возвращаемое значение - количество сформированных элементов массива.
Пример использования.
```
РАЗДЕЛИТЬ( "Раз, два, три, ", ",", "ЧастиСтрок", 0 );
// Результат - будет получен массив ЧастиСтрок с тремя элементами "Раз", "два" и "три"
```

СТРОКАИЗДАТЫ(Дата[, ФорматДаты[, РазделительДаты])
DTOC(Дата[, ФорматДаты[, РазделительДаты])
Преобразует выражение типа Дата в строку в указанном формате. Если формат преобразования не указан, то выбирается формат, соответствующий региональным

установкам. Если не указан разделитель даты, то используется разделитель, указанный в региональных настройках.
Параметры.
Дата - Дата, которая будет преобразована в строку.
ФорматДаты - Число, формат преобразования даты. Допустимые значения от 0 до 7.
0 - ДД.ММ.ГГ
1 - ММ.ДД.ГГ
2 - ГГ.ДД.ММ
3 - ГГ.ММ.ДД
4 - ДД.ММ.ГГГГ
5 - ММ.ДД.ГГГГ
6 - ГГГГ.ДД.ММ
7 - ГГГГ.ММ.ДД
РазделительДаты- Символ-разделитель. Если параметр не указан, будет использован разделитель, указанный в региональных настройках системы.
Возвращаемое значение - строка, содержащая указанную дату.

ЛЕВСИМВ(Строка, КоличествоСимволов)
LEFT(Строка, КоличествоСимволов)
Возвращает указанное количество начальных символов строки. Если КоличествоСимволов меньше или равно нулю, то будет возвращена пустая строка. Если КоличествоСимволов больше или равно длине строки, будет возвращена вся строка целиком.
Параметры.
Строка - Строка, начальные символы которой будут возвращены.
КоличествоСимволов- Число, количество символов, которое необходимо вернуть.
Возвращаемое значение - Строка, состоящая из указанного количества начальных символов исходной строки.

ДЛИНА(Строка)
LEN(Строка)
Возвращает длину указанной строки. Если строка не содержит символов, функция возвращает 0.
Параметры.
Строка - Строка, длина которой будет возвращена.
Возвращаемое значение - Число - длина, в символах, указанной строки.

СТРОЧНЫЕ(Строка)
LOWER(Строка)
Преобразует указанную строку в строчные символы.
Параметры.
Строка - Строка, которая будет преобразована в строчные символы.
Возвращаемое значение - Строка, полученная из исходной заменой всех символов на строчные.

СЖАТЬПРОБЕЛЫЛЕВ(Строка)
LTRIM(Строка)
Убирает все лидирующие пробелы из переданной строки.
Параметры.
Строка - Строка, из которой необходимо убрать пробелы.
Возвращаемое значение - Строка, полученная из исходной, без лидирующих пробелов.

ДОБАВИТЬПРОБЕЛЫЛЕВ(Строка, Длина)
PADL(Строка, Длина)
Добавляет в начало переданной строки пробелы до достижения требуемой длины. Если длина переданной строки больше требуемой, то строка будет усечена до требуемой длины.
Параметры.

Строка - Строка, к которой будут добавлены пробелы.
Длина - Число, требуемая длина результата.
Возвращаемое значение - Строка требуемой длины.

ДОБАВИТЬПРОБЕЛЫПРАВ(Строка, Длина)
PADR(Строка, Длина)
Добавляет в конец переданной строки пробелы до достижения требуемой длины. Если длина переданной строки больше требуемой, то строка будет усечена до требуемой длины.
Параметры.
Строка - Строка, к которой будут добавлены пробелы.
Длина - Число, требуемая длина результата.
Возвращаемое значение - Строка требуемой длины.

ОБРАТНЫЙПОИСК(Строка, Символ)
RAT(Строка, Символ)
Возвращает номер позиции указанного Символа в Строке. Поиск осуществляется с конца строки. Если Символ не найден, то возвращается 0.
Параметры.
Строка - Строка, в которой осуществляется поиск.
Символ - Искомый символ или Число с кодом искомого символа.
Возвращаемое значение - Число целое, позиция искомого символа в строке. Если символ не найден, то возвращается 0.

КОПИРОВАТЬ(Символ|Строка, КоличествоПовторов)
REPLICATE(Символ|Строка, КоличествоПовторов)
Функция возвращает строку, состоящую из указанного количества повторений Символа/Строки.
Параметры.
Символ - Символ, который необходимо повторить.
Строка - Строка, которую необходимо повторить.
КоличествоПовторов- Количество раз, которое необходимо произвести повторение.
Возвращаемое значение - Строка, состоящая из необходимого количества повторений Символа или Строки.

ПРАВСИМВ(Строка, Длина)
RIGHT(Строка, Длина)
Возвращает указанное количество конечных символов строки. Если КоличествоСимволов меньше или равно нулю, то будет возвращена пустая строка. Если КоличествоСимволов больше или равно длине строки, будет возвращена вся строка целиком.
Параметры.
Строка - Строка, конечные символы которой будут возвращены.
КоличествоСимволов- Число, количество символов, которое необходимо вернуть.
Возвращаемое значение - Строка, состоящая из указанного количества конечных символов исходной строки.

СЖАТЬПРОБЕЛЫПРАВ(Строка)
RTRIM(Строка)
Убирает все завершающие пробелы из переданной строки.
Параметры.
Строка - Строка, из которой необходимо убрать пробелы.
Возвращаемое значение - Строка, полученная из исходной, без завершающих пробелов.

ПРОБЕЛЫ(Количество)
SPACE(Количество)
Возвращает строку, состоящую из указанного количества пробелов.
Параметры.

Количество - Число, количество требуемых пробелов.
Возвращаемое значение - Строка, состоящая из требуемого количества пробелов.

СИМВОЛПОКОДУ(КодСимвола)
CHR(КодСимвола)
Возвращает символ, соответствующий указанному коду.
Параметры.
КодСимвола - Число целое от 0 до 255.
Возвращаемое значение - Строка, содержащая один символ, соответствующий указанному коду.

КОДСИМВОЛА(Символ)
ASC(Символ)
Возвращает код указанного символа.
Параметры.
Символ - Строка, код первого символа которой требуется вернуть.
Возвращаемое значение - Число, равное коду первого символа указанной строки.

СТРОКА(Число|ДвоичныйМассив[, Длина= 10 [,КоличествоЗнаков= 0 [,СжиматьПробелы= true [,УбиратьНули= false]]]])
STR(Число|ДвоичныйМассив[, Длина= 10 [,КоличествоЗнаков= 0 [,СжиматьПробелы= true [,УбиратьНули= false[,СистемаСчисления= 10]]]]])
Преобразует указанное число в строку указанной длины и с указанным количеством знаков после запятой.
Параметры.
Число - Число, которое необходимо преобразовать в строку.
ДвоичныйМассив - Если указан двоичный массив, то функция вернет представление массива в шестнадцатеричном виде (0x01A2B3 …)
Длина - Число, требуемая длина результирующей строки. Если параметр не указан, то возвращается строка длиной 10 символов.
КоличествоЗнаков - Число, требуемое количество знаков после запятой. Если параметр не указан, то возвращается только целая часть числа.
СжиматьПробелы- Логическое. Если Истина, то лидирующие пробелы не выводятся.
УбиратьНули - Логическое, если Истина, то завершающие нули дробной части не выводятся.
СистемаСчисления- число от 2 до 36. Позволяет преобразовать число в нужной системе счисления.
Возвращаемое значение - Строка, содержащая символьное представление числа.

STDA(Строка, Символ)
Преобразует указанную строку к виду, в котором ее можно безошибочно передать вычислителю. Пример. Необходимо передать вычислителю строку вида \"ИмяФайла = '\" + ИмяФайла + \"'\". Переменная ИмяФайл может содержать символы ' (одиночная кавычка). И напрямую такая строка вызовет ошибку при передаче ее функции Вычислить. Поэтому следует использовать такую конструкцию: \"ИмяФайла = '\" + STDA(ИмяФайла, \"'\") + \"'\". В таком виде строка безошибочно может быть передана функции Вычислить.
Параметры.
Строка - Строка, которую необходимо преобразовать.
Символ - Символ, который необходимо заменить.
Возвращаемое значение - Строка, преобразованная к безошибочному виду.

STDE(Строка[,Длина])
Преобразует указанную строку к виду, в котором ее можно безошибочно записать в текстовый файл обмена данными. При преобразовании из строки удаляются символы LF (0x0A), символ CR (0x0D) заменяется на символ ¦ (0xA6), ; заменяется на символ ¤ (0xA4).
Параметры.
Строка - Строка, которую необходимо преобразовать.

Длина – Число целое, максимальная длина возвращаемой строки. Если параметр не указан, то длина не ограничивается.
Возвращаемое значение – Строка, преобразованная к безошибочному виду.

STDF(Строка[,Длина])
Преобразует указанную строку к виду, в котором ее можно безошибочно передать в SQL запросе. При преобразовании символ ' (одиночная кавычка) заменяется на '' (две одиночные кавычки), убираются лишние пробелы и длина строки результата не превышает указанной длины.
Параметры.
Строка – Строка, которую необходимо преобразовать.
Длина – Число целое, максимальная длина возвращаемой строки. Если параметр не указан, то длина не ограничивается.
Возвращаемое значение – Строка, преобразованная к безошибочному виду.

ЗАМЕНИТЬ(СтрокаПоиска,ИскомаяСтрока,Замена[,НомерФрагмента[,КоличествоЗамен]])
STRTRAN(СтрокаДляПоиска,ИскомаяСтрока,Замена[,НомерФрагмента[,КоличествоЗамен]])
Возвращает строку, состоящую из СтрокиДляПоиска, в которой все вхождения (или указанное количество) ИскомойСтроки заменены на Замену. Также есть возможность указать номер фрагмента, с которого необходимо осуществлять замену, и количество осуществляемых замен. Например, первое вхождение можно пропустить, а начать заменять со второго и осуществить только две замены. Поиск вхождения осуществляется с учетом регистра.
Параметры.
СтрокаДляПоиска – Строка, в которой будут произведены поиск и замена.
ИскомаяСтрока – Строка, которую необходимо найти и заменить.
Замена – Строка, на которую необходимо заменить найденную ИскомуюСтроку.
НомерФрагмента – Число, номер найденного фрагмента, с которого необходимо начать замены. Если параметр не указан, то замены начнутся с первого вхождения.
КоличествоЗамен – Число, количество осуществляемых замен. Если параметр не указан, то будут заменены все найденные вхождения.
Возвращаемое значение – Строка, полученная из исходной строки заменой требуемых вхождений.

STRTRANC(СтрокаПоиска,ИскомаяСтрока,Замена[,НомерФрагмента[,КоличествоЗамен]])
Возвращает строку, состоящую из СтрокиДляПоиска, в которой все вхождения (или указанное количество) ИскомойСтроки заменены на Замену. Также есть возможность указать номер фрагмента, с которого необходимо осуществлять замену, и количество осуществляемых замен. Например, первое вхождение можно пропустить, а начать заменять со второго и осуществить только две замены. Поиск вхождения осуществляется без учета регистра.
Параметры.
СтрокаДляПоиска – Строка, в которой будут произведены поиск и замена.
ИскомаяСтрока – Строка, которую необходимо найти и заменить.
Замена – Строка, на которую необходимо заменить найденную **ИскомуюСтроку.**
НомерФрагмента – Число, номер найденного фрагмента, с которого необходимо начать замены. Если параметр не указан, то замены начнутся с первого вхождения.
КоличествоЗамен – Число, количество осуществляемых замен. Если параметр не указан, то будут заменены все найденные вхождения.
Возвращаемое значение – Строка, полученная из исходной строки заменой требуемых вхождений.

ПОДСТРОКА(Строка, НачальнаяПозиция[, КоличествоСимволов])
SUBSTR(Строка, НачальнаяПозиция[, КоличествоСимволов])
Возвращает указанное количество символов строки, начиная с НачальнойПозиции. Если КоличествоСимволов меньше или равно нулю, то будет возвращена пустая

строка. Если КоличествоСимволов не указано, то будет возвращена подстрока, начиная с НачальнойПозиции до конца исходной строки.
Параметры.
Строка - Строка, из которой производится выборка подстроки.
НачальнаяПозиция - Число, номер символа в строке, с которого начинается выборка подстроки. Нумерация начинается с 1.
КоличествоСимволов- Число, количество символов, которое необходимо вернуть.
Возвращаемое значение - Строка, состоящая из указанного количества символов исходной строки начиная с НачальнойПозиции.

СТРОКАИЗВРЕМЕНИ(ДатаВремя[, ФорматДаты[, РазделительДаты[, РазделительДатыИВремени] [, РазделительВремени]]]])
ТТОС(ДатаВремя[, ФорматДаты[, РазделительДаты[, РазделительДатыИВремени] [, РазделительВремени]]]])
Преобразует выражение типа ДатаВремя в строку в указанном формате. Если формат преобразования не указан, то выбирается формат, соответствующий региональным установкам. Если не указан разделитель даты, то используется разделитель, указанный в региональных настройках.
Параметры.
ДатаВремя - Дата и время, которые необходимо преобразовать в строку.
ФорматДаты - Число, формат преобразования даты. Допустимые значения от 0 до 8.
0 - ДД.ММ.ГГ ЧЧ:ММ:СС
1 - ММ.ДД.ГГ ЧЧ:ММ:СС
2 - ГГ.ДД.ММ ЧЧ:ММ:СС
3 - ГГ.ММ.ДД ЧЧ:ММ:СС
4 - ДД.ММ.ГГГГ ЧЧ:ММ:СС
5 - ММ.ДД.ГГГГ ЧЧ:ММ:СС
6 - ГГГГ.ДД.ММ ЧЧ:ММ:СС
7 - ГГГГ.ММ.ДД ЧЧ:ММ:СС
8 - ЧЧ:ММ:СС, только время
РазделительДаты - Символ-разделитель. Если параметр не указан, будет использован разделитель, указанный в региональных настройках системы.
РазделительДатыИВремени- Символ-разделитель вставляемый между датой и временем. Если параметр не указан, то будет использован пробел.
РазделительВремени- Символ-разделитель частей времени. Если параметр не указан, то будет использовано двоеточие ':'.
Возвращаемое значение - строка, содержащая указанную дату и время.

ПРОПИСНЫЕ(Строка)
UPPER(Строка)
Преобразует указанную строку в прописные (заглавные) символы.
Параметры.
Строка - Строка, которая будет преобразована в прописные символы.
Возвращаемое значение - Строка, полученная из исходной заменой всех символов на прописные.

ПРОЧИТАТЬСТРОКУ(ИмяПеременной[,Разделитель])
STRINGREAD(ИмяПеременной[,Разделитель])

Возвращает первую строку из значения указанной переменной, оканчивающуюся разделителем. Если разделитель не указан, то строка читается до стандартного разделителя 0x0A. При этом символ 0x0D всегда просто пропускается и не добавляется в возвращаемую строку. Если указан разделитель, то строка читается строго до него. Возвращаемая строка не содержит разделителя. Оставшийся текст без первой строки записывается в указанную переменную.
Параметры.
ИмяПеременной- Строка, имя переменной, значение которой считается текстом.
Разделитель - Строка, ограничитель строки.

Возвращаемое значение - Первая строка текста, не включающая разделитель. Значение переменной из первого параметра будет изменено на текст без первой строки ограничителей.

ПОКРУЖЕНИЯ(ИмяПеременнойОкружения)
ENVVAR(ИмяПеременнойОкружения)
Возвращает строку — значение переменной окружения, если такая определена. Если указанной переменной нет в окружении, то пустую строку.
Параметры.
ИмяПеременнойОкружения - Строка, имя переменной окружения.
Возвращаемое значение - строка. Значение указанной переменной.

ПОЛУЧИТЬСЛОВО(Строка[,ОграничителиНачала[,ОграничителиКонца[, НомерСлова[,НачальнаяПозиция]]]])
GETWORD(Строка[,ОграничителиНачала[,ОграничителиКонца[,НомерСлова[, НачальнаяПозиция]]]])
Возвращает указанное слово из строки. Словом называется произвольная часть строки, ограниченная с начала символами из параметра ОграничителиНачала и с конца символами из параметра ОграничителиКонца. Если параметры ОграничителиНачала и ОграничителиКонца не указаны, или указаны пустые строки, то ограничителями слова являются любые символы не являющиеся буквой, цифрой или специальными символами: @ _ - . Параметр НомерСлова позволяет получить не первое слово, а указанное. Указание параметра НачальнаяПозиция позволяет осуществить поиск слова не с начала Строки, а с указанной позиции.
Параметры.
Строка - Строка, в которой будет осуществлен поиск слова.
Ограничители - Строки, которые должны содержать символы ограничители слова.
НомерСлова - Число, номер слова, которое необходимо найти.
НачальнаяПозиция - Число, номер позиции в исходной строке, с которой необходимо осуществить поиск.
Возвращаемое значение - Строка, содержащая указанное слово.
Пример. **ПОЛУЧИТЬСЛОВО**('Съешь же еще этих мягких [французских] булок, [да] выпей чаю.', '[',']', 2). Результат – да

ПРОПИСЬ(Число, Слово1, Слово2, Слово5[, КоличествоЗнаков[, МужЖенСредРод]])
TRANSLATENUMBER(Число, Слово1, Слово2, Слово5[, КоличествоЗнаков[, МужЖенСредРод]])
Возвращает указанное число прописью и добавляет одно из указанных Слов в конце строки. Слова должны соответствовать формам, используемым с числительными, оканчивающимся, соответственно, на 1, кроме 11, на 2, 3 или 4, кроме 12,13,14 и на 5,6,7,8,9,11,12,13,14. Если указан параметр КоличествоЗнаков, то после Слов будет выведена дробная часть в виде цифр. Для корректного вывода числительных один, одно, одна, два и две необходимо указать параметр МужЖенСредРод.
Параметры.
Число - Число, которое будет написано прописью.
Слово1 - Строка, именительный падеж единственного числа Слова. Будет добавлено к числительным, оканчивающимся на 1, кроме 11.
Слово2 - Строка, родительный падеж единственного числа Слова. Будет добавлено к числительным, оканчивающимся на 2, 3 или 4, кроме 12, 13 и 14.
Слово5 - Строка, родительный падеж множественного числа Слова. Будет добавлено к числительным, оканчивающимся на 5, 6, 7, 8, 9, 11, 12, 13, 14 и 15.
КоличествоЗнаков- Число, количество знаков после запятой, которое будет выведено после Слов в виде цифр.
МужЖенСредРод- Число, указывает в каком роде выводить числительные один и два. 0- мужской род, 1- женский, 2- средний.
Возвращаемое значение - Строка, содержащая указанное число прописью.
Пример. **ПРОПИСЬ**(1234.56, 'стол', 'стола', 'столов', 2, 1). Результат - одна тысяча двести тридцать четыре стола 56

ПРОПИСЬРУБ(Число[, КоличествоЗнаков])
TRANSLATENUMBERRUR(Число[, КоличествоЗнаков])
Возвращает указанное количество рублей прописью. Функция является упрощенной формой функции **ПРОПИСЬ**.
Параметры.
Число - Число, которое будет написано прописью. После числа прописью будет добавлено одно из слов 'рубль', 'рубля' или 'рублей'.
КоличествоЗнаков - Число, количество копеек, которое будет выведено после Слов в виде цифр и слова 'копейка', 'копейки' или 'копеек'.
Возвращаемое значение - Строка, содержащая указанное количество рублей прописью.
Пример. **ПРОПИСЬРУБ**(1234.56, 2), Результат - одна тысяча двести тридцать четыре рубля 56 копеек

ПЕРЕКОДИРОВАТЬ(Строка|ЮникодМассив,ИсходнаяКодировка,РезультирующаяКодировка)
TRANSLATE(Строка|ЮникодМассив,ИсходнаяКодировка,РезультирующаяКодировка)
Функция перекодирует указанную строку однобайтовых символов или массив символов в формате Unicode в строку однобайтовых символов в другой кодировке или в массив символов в формате Unicode. Если исходная кодировка является однобайтовой кодировкой, то первый параметр должен быть строкой однобайтовых символов, иначе - массивом символов в формате Unicode. Если результирующая кодировка является однобайтовой кодировкой, то функция возвращает строку однобайтовых символов, иначе - двоичный массив символов в формате Unicode.
Параметры.
Строка - Строка однобайтовых символов, которую надо перекодировать.
МассивСимволов - Двоичный массив символов в формате Unicode, который надо перекодировать в строку.
ИсходнаяКодировка - Число целое или строка. Допустимые значения для данного параметра:
Строка 'ANSI' - ANSI кодировка,
Строка 'OEM' - OEM кодировка,
Строка 'UTF-8' - UTF-8 кодировка,
Строка 'UTF-16' или 'Unicode' - UTF-16 кодировка,
Число целое - код кодировки.
РезультирующаяКодировка- Число целое или строка.
Допустимые значения для данного параметра:
Строка 'ANSI' - ANSI кодировка,
Строка 'OEM' - OEM кодировка,
Строка 'UTF-8' - UTF-8 кодировка,
Строка 'KOI8' - кодировка KOI8,
Строка 'UTF-16' или 'Unicode' - Unicode кодировка,
Число целое - код кодировки.
Возвращаемое значение - Строка, если результирующая кодировка однобайтовая, иначе двоичный массив.

СТРОКАИЗМАССИВА(МассивБайт)
ARRAYTOSTRING(МассивБайт)
Функция преобразует массив байт в строку, состоящую из символов массива байт.
Параметры.
МассивБайт- массив, который необходимо преобразовать в строку.
Возвращаемое значение - строка, полученная из массива байт.

МАССИВИЗСТРОКИ(Строка)
STRINGTOARRAY(Строка)
Функция возвращает массив байт, полученный из символов строки без каких либо преобразований.
Параметры.
Строка- строка, которую необходимо преобразовать в массив байт.
Возвращаемое значение - массив байт, полученный из строки.

ЭЛЕМЕНТ(Строка | Массив, Индекс[, НовоеЗначение])
ITEM(Строка | Массив, Индекс[, НовоеЗначение])
Функция возвращает указанный элемент строки или массива в виде целого числа. Если задано новое значение, то оно будет присвоено элементу массива или строки.
Параметры.
Строка|Массив- строка или массив, из которых необходимо выбрать элемент. Если указано новое значение элемента, то параметр должен содержать имя переменной.
Индекс- индекс получаемого / устанавливаемого элемента.
НовоеЗначение- число целое или символ для установки значения.
Возвращаемое значение - число целое - значение указанного элемента строки или массива. При замене элемента возвращается предыдущее значение.

ЗАШИФРОВАТЬТЕКСТ(Текст, Ключ)
ENCRYPTTEXT(Текст, Ключ)
Возвращает зашифрованный текст, указанным ключом. Результат шифрования сразу кодируется в BASE64.
Параметры.
Текст - Строка, требующая шифрования.
Ключ - Строка, ключ для шифрования.
Возвращаемое значение - строка, зашифрованный текст, преобразованный в base64.

РАСШИФРОВАТЬТЕКСТ(Текст, Ключ)
DECRYPTTEXT(Текст, Ключ)
Возвращает расшифрованный текст, указанным ключом. Входящий текст должен быть в кодировке base64, например, возвращенный функцией **ЗАШИФРОВАТЬТЕКСТ**( ).
Параметры.
Текст - Строка, зашифрованный текст в кодировке base64.
Ключ - Строка, ключ для расшифровки.
Возвращаемое значение - строка, расшифрованный текст.

ФУНКЦИИ ДЛЯ РАБОТЫ С ДАТАМИ

ДАТАИЗВРЕМЕНИ(ДатаВремя)
TTOD(ДатаВремя)
Преобразует указанную дату и время в дату без времени.
Параметры.
ДатаВремя - Дата и время, которые необходимо преобразовать в Дату.
Возвращаемое значение - ДатаВремя, полученное из исходного значения, но с полем Время, равным 0.

ИМЯДНЯНЕДЕЛИ(Дата|НомерДня)
CDOW(Дата|НомерДня)
Возвращает имя дня недели по указанной дате или по номеру дня недели.
Параметры.
Дата - Дата, для которой рассчитывается имя дня недели.
НомерДня - Число, номер дня недели, имя которого необходимо определить.
Возвращаемое значение - Строка, содержащая имя дня недели. Понедельник - первый день.

ДАТАИЗСТРОКИ(Строка|Число)
CTOD(Строка|Число)
Преобразует указанную строку или число в дату.
Параметры.
Строка- Строка, содержащая дату в формате ДД.ММ.ГГГГ ЧЧ:ММ:СС.
Число - Номер дня по Юлианскому календарю, который надо преобразовать в дату.
Возвращаемое значение - ДатаВремя. Если указанная строка не может быть преобразована в дату, то будут возвращены пустые дата и время.

ДАТА()
DATE()
Возвращает текущую системную дату (без времени).
Параметры.
нет.
Возвращаемое значение - ДатаВремя, содержащее текущую системную дату без времени.

ДАТАВРЕМЯ()
DATETIME()
Возвращает текущую системную дату и время.
Параметры.
нет.
Возвращаемое значение - ДатаВремя, содержащее текущие системную дату и время.

ДЕНЬ(Дата)
DAY(Дата)
Возвращает номер дня месяца из указанной даты.
Параметры.
Дата - Дата, для которой рассчитывается номер дня месяца.
Возвращаемое значение - Число, соответствующее номеру дня месяца указанной даты.

ДЕНЬНЕДЕЛИ(Дата)
DOW(Дата)
Возвращает номер дня недели по указанной дате.
Параметры.
Дата - Дата, для которой рассчитывается номер дня недели.
Возвращаемое значение - Число, соответствующее номеру дня недели указанной даты.

ДОБАВИТЬДНИ(Дата[, КоличествоДней= 1])
GODAY(Дата[, КоличествоДней= 1])
Возвращает дату, отстоящую от указанной даты на указанное количество дней. Если КоличествоДней больше нуля, то полученная дата будет позже исходной. Если КоличествоДней меньше нуля, то полученная дата будет раньше исходной.
Параметры.
Дата - Дата, от которой производится расчет.
КоличествоДней - Число, количество дней, которое необходимо прибавить к указанной дате. Если параметр не указан, то прибавляется 1 день.
Возвращаемое значение - Дата, отстоящая от указанной на указанное количество дней.

ДОБАВИТЬМЕСЯЦЫ(Дата[, КоличествоМесяцев= 1])
GOMONTH(Дата[, КоличествоМесяцев= 1])
Возвращает дату, отстоящую от указанной даты на указанное количество месяцев. Если КоличествоМесяцев больше нуля, то полученная дата будет позже исходной. Если КоличествоМесяцев меньше нуля, то полученная дата будет раньше исходной.
Параметры.
Дата - Дата, от которой производится расчет.
КоличествоМесяцев - Число, количество месяцев, которое необходимо прибавить к указанной дате. Если параметр не указан, то прибавляется 1 месяц.
Возвращаемое значение - Дата, отстоящая от указанной на указанное количество месяцев.

ДОБАВИТЬГОДЫ(Дата[, КоличествоЛет= 1])
GOYEAR(Дата[, КоличествоЛет= 1])
Возвращает дату, отстоящую от указанной даты на указанное количество лет. Если КоличествоЛет больше нуля, то полученная дата будет позже исходной. Если КоличествоЛет меньше нуля, то полученная дата будет раньше исходной.
Параметры.

Дата - Дата, от которой производится расчет.
КоличествоЛет - Число, количество лет, которое необходимо прибавить к указанной дате. Если параметр не указан, то прибавляется 1 год.
Возвращаемое значение - Дата, отстоящая от указанной на указанное количество лет.

РАЗНИЦАДАТ(Дата1, Дата2, СравниваемаяЧасть)
DATEDIFF(Дата1, Дата2, СравниваемаяЧасть)
Вычисляет разницу между двумя указанными значениями с типом ДатаВремя. Третий параметр указывает в каких единицах вычисляется разница - в секундах, минутах, часах, днях, месяцах или годах. Расчет производится вычитанием первого параметра **Дата1** из второго **Дата2**.
Параметры.
Дата1 - ДатаВремя, первая дата и время, участвующая в операции.
Дата2 - ДатаВремя, вторая дата и время, участвующая в операции.
СравниваемаяЧасть - Строка, указывающая, в каких единицах необходимо рассчитывать разницу.

- **'sec'** - в секундах,
- **'min'** - в минутах,
- **'hour'** - в часах,
- **'day'** - в днях,
- **'mon'** - в месяцах,
- **'year'** - в годах.

Возвращаемое значение - Число целое, указывающее разницу в значениях в выбранных единицах измерения.

ЧАС(ДатаВремя)
HOUR(ДатаВремя)
Возвращает номер часа из указанных даты и времени.
Параметры.
ДатаВремя - Дата и время, для которых функция вернет значение.
Возвращаемое значение - Число целое от 0 до 23, номер часа из указанных даты и времени.

МИНУТА(ДатаВремя)
MINUTE(ДатаВремя)
Возвращает значение минут из указанных даты и времени.
Параметры.
ДатаВремя - Дата и время, для которых функция вернет значение.
Возвращаемое значение - Число целое от 0 до 59 значение минут из указанных даты и времени.

СЕКУНДА(ДатаВремя|Секунды)
SECOND(ДатаВремя|Секунды)
Возвращает значение секунд из указанных даты и времени.
Параметры.
ДатаВремя - Дата и время, для которых функция вернет значение.
Секунды - Число секунд, для расчета количества секунд.
Возвращаемое значение - Число целое от 0 до 59, значение секунд из указанных даты и времени или числа секунд.

МЕСЯЦ(Дата)
MONTH(Дата)
Возвращает номер месяца из указанной даты.
Параметры.
Дата - Дата, для которой функция вернет значение.
Возвращаемое значение - Число целое от 1 до 12, номер месяца из указанной даты.

ИМЯМЕСЯЦА(Дата|НомерМесяца[, Параметр2])
MONTHNAME(Дата|НомерМесяца[, Параметр2])
Возвращает название месяца из указанной даты или по указанному номеру месяца.
Параметры.
Дата – Дата, для которой функция вернет значение.
НомерМесяца – Номер месяца от 1 до 12, для которого функция вернет значение.
Параметр2 – Число. Если значение параметра равно нулю, то будет возвращено название месяца в Именительном падеже. Иначе – в Родительном падеже. Если параметр не указан, то используется значение 0.
Возвращаемое значение – Строка, содержащая название месяца в указанном падеже.

ВРЕМЯ()
TIME()
Возвращает текущее системное время.
Параметры.
нет.
Возвращаемое значение – ДатаВремя, содержащее текущее системное время. Поле Дата возвращаемого значения содержит 0.

НЕДЕЛЯ(Дата)
WEEK(Дата)
Возвращает номер недели в году по указанной дате.
Параметры.
Дата – Дата, для которой рассчитывается номер недели.
Возвращаемое значение – Число, соответствующее номеру недели в году для указанной даты.

ГОД(Дата)
YEAR(Дата)
Возвращает год из указанной даты.
Параметры.
Дата – Дата, для которой возвращается год.
Возвращаемое значение – Число, соответствующее номеру года для указанной даты.

КВАРТАЛ(Дата)
QUARTER(Дата)
Возвращает номер квартала в году по указанной дате.
Параметры.
Дата – Дата, для которой рассчитывается номер квартала.
Возвращаемое значение – Число, соответствующее номеру квартала в году для указанной даты.

НАЧАЛОМЕСЯЦА(Дата)
MONTHFIRSTDATE(Дата)
Возвращает дату, соответствующую началу месяца для указанной даты.
Параметры.
Дата – Дата, для которой рассчитывается дата начала месяца.
Возвращаемое значение – Дата, соответствующая началу месяца для указанной даты.

КОНЕЦМЕСЯЦА(Дата)
MONTHLASTDATE(Дата)
Возвращает дату, соответствующую последнему дню месяца для указанной даты.
Параметры.
Дата – Дата, для которой рассчитывается последний день месяца.
Возвращаемое значение – Дата, соответствующая последнему дню месяца для указанной даты.

ФУНКЦИИ ДЛЯ РАБОТЫ С ЧИСЛАМИ

АБСОЛЮТНОЕЗНАЧЕНИЕ(Число)
ABS(Число)
Функция вычисляет и возвращает абсолютное значение своего аргумента. Аргумент должен быть либо целым либо вещественным числом.
Параметры.
Число – Число целое или вещественное, абсолютное значение которого необходимо вычислить
Возвращаемое значение – число, равное абсолютному значению аргумента функции.

ЭЛЕМЕНТОВВМАССИВЕ(ИмяМассива)
ACOUNT(ИмяМассива)
Функция возвращает количество элементов в указанном массиве.
Параметры.
ИмяМассива – Строка (в кавычках) имя массива.
Возвращаемое значение – число целое, равное количеству элементов в массиве. Если указанного массива нет, то вернется значение 0.

ЦЕЛОЕ(Число)
INT(Число)
Возвращает целую часть числа.
Параметры.
Число – Число, целое или вещественное, целая часть которого будет возвращена.
Возвращаемое значение – целое, целая часть указанного Числа.

ОКРУГЛИТЬ(Число, КоличествоЗнаков)
ROUND(Число, КоличествоЗнаков)
Возвращает Число, округленное до указанного количества знаков до или после запятой. Если указанное количество знаков больше нуля, то округление производится до соответствующих долей числа. Если указанное количество знаков меньше нуля, то округление производится до, соответственно, десятков, сотен и т.д.
Параметры.
Число – Число, требующее округления.
КоличествоЗнаков – Число, количество знаков до или после запятой, до которого необходимо произвести округление.
Возвращаемое значение – Число вещественное.

ЗНАК(Число)
SIGN(Число)
Функция возвращает 1, если указанное Число положительное, -1, если оно отрицательное и 0, если число равно нулю.
Параметры.
Число – Число, знак которого необходимо определить.
Возвращаемое значение – Число целое.

СЛУЧАЙНОЕЧИСЛО([Параметр])
RAND([Аргумент])
Возвращает псевдослучайное число. Если Параметр не указан, то будет возвращено следующее псевдослучайное число. Если в качестве параметра передано число, то генератор псевдослучайных чисел будет инициализирован этим числом. Если передан параметр любого другого типа, например, логический, то генератор псевдослучайных чисел будет инициализирован текущими системными датой и временем.
Параметры.
Параметр – Число, база для инициализации генератора псевдослучайных чисел.
Возвращаемое значение – Число вещественное, псевдослучайное число.

VAL(Строка)

Преобразует указанную строку в числовое значение.
Параметры.
Строка - Строка, которая будет преобразована в число.
Возвращаемое значение - Число. Если в исходной строке есть десятичная точка или запятая, то результатом будет Число вещественное. Иначе - Число целое.

ЛОГИЧЕСКИЕ ФУНКЦИИ

МЕЖДУ(Выражение1, Выражение2, Выражение3)
BETWEEN(Выражение1, Выражение2, Выражение3)
Вычисляет, находится ли Выражение1 между Выражением2 и Выражением3
Параметры.
Выражение1 - Число, Строка или ДатаВремя.
Выражение2 - Выражение, тип которого должен совпадать с типом Выражения1.
Выражение3 - Выражение, тип которого должен совпадать с типом Выражения1
Возвращаемое значение - логическое. Истина, если выполняется неравенство Выражение2 <= Выражение1 <= Выражение3. Иначе - Ложь.

ПУСТО(Выражение)
EMPTY(Выражение)
Возвращает ИСТИНУ, если указанное выражение пустое. Строка считается пустой, если она не содержит символов или коды всех символов не превосходят 32 (код пробела). Число считается пустым, если оно равно 0. Дата считается пустой, если ее числовое представление равно 0. Логическое выражение считается пустым, если его значение равно **ЛОЖЬ**.
Параметры.
Выражение- Выражение, значение которого будет проанализировано.
Возвращаемое значение - логическое. **Истина**, если значение выражения считается пустым. Иначе - **Ложь**.

ЕСЛИ(Условие, Выражение1, Выражение2)
IIF(Условие, Выражение1, Выражение2)
Функция возвращает одно из двух значений, в зависимости от истинности указанного Условия. Если Условие истинно, то возвращается значение Выражения1. Иначе - значение Выражения2.
Параметры.
Условие - Логическое выражение, значение которого используется при выборе возвращаемого значения.
Выражение1 - Выражение любого типа, значение которого будет возвращено, если Условие истинно.
Выражение2 - Выражение любого типа, значение которого будет возвращено, если Условие ложно.
Возвращаемое значение - зависит от значения Условия и указанных Выражений.

ВСПИСКЕ(Выражение1, Выражение2[,...])
INLIST(Выражение1, Выражение2[,...])
Проверяет наличие указанного значения в списке значений. Размер указанного списка значений неограничен.
Параметры.
Выражение1 - Выражение любого типа, значение которого будет искаться в списке.
Выражение2,... - Список выражений, среди значений которых будет осуществлен поиск.
Возвращаемое значение - логическое. Истина, если значение Выражения1 находится в списке значений Выражение2,...,ВыражениеN.

МАКС(Выражение1, Выражение2[,Выражение3[,...]])
MAX(Выражение1, Выражение2[,Выражение3[,...]])

Возвращает максимальное из двух и более значений.
Параметры.
Выражение1 - Выражение любого типа, значение которого будет участвовать в сравнении.
Выражение2 - Выражение того же типа, что и Выражение1, значение которого будет участвовать в сравнении.
Возвращаемое значение - максимальное из значений указанных выражений.

МИН(Выражение1, Выражение2[,Выражение3[,...]])
MIN(Выражение1, Выражение2[,Выражение3[,...]])
Возвращает минимальное из двух и более значений.
Параметры.
Выражение1 - Выражение любого типа, значение которого будет участвовать в сравнении.
Выражение2 - Выражение того же типа, что и Выражение1, значение которого будет участвовать в сравнении.
Возвращаемое значение - минимальное из значений указанных выражений.

ФУНКЦИИ ДЛЯ РАБОТЫ С СОЕДИНЕНИЯМИ ODBC

ДОБАВИТЬСОЕДИНЕНИЕ(ИмяСоединения | СтрокаСоединения)
ADDCONNECTION(ИмяСоединения | СтрокаСоединения)
Функция производит подключение к указанному источнику данных по протоколу ODBC.
Параметры.
ИмяСоединения - Строка, имя соединения из таблицы qreport_connections.
СтрокаСоединения - Строка, описывающая подключение к источнику данных в формате ODBC.
Возвращаемое значение - Число целое. Если значение больше или равно 0, то дескриптор соединения, иначе - ошибка подключения.
Пример строки соединения с базой данных Oracle:
DSN=Oracle;UID=login;PWD=password;DBQ=OracleServiceName;DBA=W;APA=T;FEN=T;QTO=T;FRC=10;FDL=10;LOB=T;RST=T;FRL=F;MTS=F;CSR=F;PFC=10;TLO=0;
Пример строки соединения с базой данных MSSQL:
DRIVER=SQL Native Client;SERVER=(local)\sqlexpress;UID=idleadmin;PWD=itida;DATABASE=itidaretail;AutoTranslate=No

РАЗОРВАТЬСОЕДИНЕНИЕ(ДескрипторСоединения)
BREAKCONNECTION(ДескрипторСоединения)
Функция производит завершение соединения с источником данных, которое было установлено функцией ДобавитьСоединение( ). При этом освобождаются все занятые соединением ресурсы.
Параметры.
ДескрипторСоединения - Число целое, дескриптор завершаемого соединения.
Возвращаемое значение - нет."

ВЫБРАТЬСОЕДИНЕНИЕ(ДескрипторСоединения)
SELECTCONNECTION(ДескрипторСоединения)
Функция устанавливает указанное соединение соединением по умолчанию. При этом все функции выполняющие SQL запросы будут передавать их указанному соединению, если явным образом не указано иное. Если функции передан несуществующий дескриптор соединения, например -1, то функция выберет основное соединение в качестве текущего.
Параметры.
ДескрипторСоединения - Число целое, дескриптор выбираемого соединения.
Возвращаемое значение - Число целое, дескриптор ранее выбранного соединения или -1, если ранее не было выбрано никакое соединение.

СОЕДИНЕНИЕ([НомерСоединения])
CONNECTION([НомерСоединения])

Возвращает дескриптор указанного соединения с базой данных.
Параметры.
НомерСоединения- Число целое, номер соединения, который вернула функция **ДОБАВИТЬСОЕДИНЕНИЕ**. Если параметр не указан, то будет возвращен дескриптор текущего соединения с базой данных.
Возвращаемое значение - Число целое, дескриптор указанного соединения\nc базой данных.

ФУНКЦИИ ДЛЯ РАБОТЫ С ДАННЫМИ

ЗАПРОС(SQLЗапрос[,Поле,[НомерСоединения|СписокФилиалов]])
QUERY(SQLЗапрос[,Поле,[НомерСоединения | СписокФилиалов]])
Выполняет указанный запрос SQL и возвращает значение указанного поля первой записи результата выполнения запроса. Если имя поля не указано, то возвращается значение первого поля первой записи результата выполнения запроса. Если сервер вернул ошибку выполнения запроса, то функция возвращает Ложь и записывает полученное сообщение от сервера в журнал сообщений и ошибок (errorlog).
Параметры.
SQLЗапрос - Строка, содержащая SQL запрос к источнику данных.
Поле - Строка, имя поля результата выполнения запроса, значение которого необходимо вернуть. Если параметр не указан, будет возвращено значение первого поля.
НомерСоединения- позволяет указать номер источника данных, в котором будет выполнен запрос. Данный номер возвращается функцией ДобавитьСоединение.
СписокФилиалов- Список филиалов через пробел, в базах которых будет выполнен SQL запрос.
Возвращаемое значение - зависит от типа полученных данных.

СФОРМИРОВАТЬ_XML([Описание[,Описание[,...]]])
GETXML([Описание[,Описание[,...]]])
Функция формирует текст в формате XML по указанным описаниям. Если описания не указаны, то текст формируется на основании текущей строки выбранного контекста. В качестве тэгов в формируемом тексте выступают имена полей результата выполнения запроса из описания, либо из текущего контекста, если описания не указаны. Описаний может быть несколько, тогда считается, что каждое последующее описание связано с предыдущим указанным условием.
Параметры.
Описание - Строка описание выгружаемых данных. Строка должна быть в формате XML и может содержать следующие тэги:
<**ЗАПРОС**> или <**QUERY**> - Строка, вычисление которой должно возвращать SQL запрос, для формирования таблицы с данными.
<**ТЭГЗАГОЛОВКА**> или <**HEADERTAG**> - тэг заголовка таблицы с данными.
<**ТЭГСТРОКИ**> или <**ROWTAG**> - Строка, тэг разделитель строк.
<**ИМЯКОНТЕКСТА**> или <**CONTEXTNAME**> - Строка, имя контекста для загрузки данных и использования в подчиненных описаниях.
<**ТЕКУЩАЯСТРОКА**> или <**CURRENTROW**> - Логическое, если ИСТИНА, то XML текст будет сформирован только по текущей строке указанного контекста.
<**ОТСТУП**> или <**INDENT**> - Число целое, отступ, с которым необходимо выводить данные.
<**АТРИБУТ_ИмяТэга**> или <**ATTRIBUTE_ИмяТэга**> - Строка, дополнительные атрибуты тэга.
<**ВЫЧИСЛЯЕМЫЙАТРИБУТ_имя тэга**> или <**EVALUATEATTRIBUTE_имя тэга**> - Строка, выражение для вычисления дополнительного атрибута тэга.
<**ИНДИКАТОР**> или <**PROGRESSBAR**> - Число, шаг обновления индикатора. При указании положительного шага, функция будет передавать сообщение главному окну для обновления индикатора хода работы.
Возвращаемое значение - Строка, содержащая требуемые данные в формате XML.
Пример.

СФОРМИРОВАТЬ_XML(\"<ЗАПРОС>'SELECT field1, field2, field3 FROM table1 WHERE key='+key</ЗАПРОС><ТЭГЗАГОЛОВКА>Table</ТЭГЗАГОЛОВКА><ТЭГСТРОКИ>Row</ТЭГСТРОКИ><ИМЯКОНТЕКСТА>table1</ИМЯКОНТЕКСТА>\", \"<ЗАПРОС>'SELECT field1, field2, field3 FROM table2 WHERE key2='+table1.field1</ЗАПРОС><ТЭГЗАГОЛОВКА/><ТЭГСТРОКИ>subrow<ТЭГСТРОКИ>\")

СФОРМИРОВАТЬ_JSON([Описание[,Описание[,...]]])
GETJSON([Описание[,Описание[,...]]])
Функция формирует текст в формате JSON по указанным описаниям. Если описания не указаны, то текст формируется на основании текущей строки выбранного контекста. В качестве тэгов в формируемом тексте выступают имена полей результата выполнения запроса из описания, либо из текущего контекста, если описания не указаны. Описаний может быть несколько, тогда считается, что каждое последующее описание связано с предыдущим указанным условием.
Параметры.
Описание - Строка описание выгружаемых данных. Строка должна быть в формате XML и может содержать следующие тэги:
<ЗАПРОС> или **<QUERY>** - Строка, вычисление которой должно возвращать SQL запрос, для формирования таблицы с данными.
<ТЭГЗАГОЛОВКА> или **<HEADERTAG>** - тэг заголовка таблицы с данными.
<ТЭГСТРОКИ> или **<ROWTAG>** - Строка, тэг разделитель строк.
<ИМЯКОНТЕКСТА> или **<CONTEXTNAME>** - Строка, имя контекста для загрузки данных и использования в подчиненных описаниях.
<ТЕКУЩАЯСТРОКА> или **<CURRENTROW>** - Логическое, если ИСТИНА, то JSON текст будет сформирован только по текущей строке указанного контекста.
<ОТСТУП> или <**INDENT>** - Число целое, отступ, с которым необходимо выводить данные.
<ИНДИКАТОР> или **<PROGRESSBAR>** - Число, шаг обновления индикатора. При указании положительного шага, функция будет передавать сообщение главному окну для обновления индикатора хода работы.
<ПРОПУСТИТЬ> или **<SKIP>** - Строка, имя поля, которое необходимо пропустить (не выводить).
<ВЛАДЕЛЕЦ> или **<MASTER>** - Строка, имя контекста, которому подчиняется текущая группа.
<РЕКУРСИЯ> или **<RECURSION>** - Строка, содержащая условие, выполнение которого вызывает рекурсию текущей группы.
Возвращаемое значение - Строка, содержащая требуемые данные в формате JSON.
Пример.
СФОРМИРОВАТЬ_JSON("<ЗАПРОС>'SELECT field1, field2, field3 FROM table1 WHERE key='+key</ЗАПРОС><ТЭГЗАГОЛОВКА>Table</ТЭГЗАГОЛОВКА><ТЭГСТРОКИ>Row</ТЭГСТРОКИ><ИМЯКОНТЕКСТА>table1</ИМЯКОНТЕКСТА>", "<ЗАПРОС>'SELECT field1, field2, field3 FROM table2 WHERE key2='+ table1.field1</ЗАПРОС><ТЭГЗАГОЛОВКА/><ТЭГСТРОКИ>subrow<ТЭГСТРОКИ>").

СФОРМИРОВАТЬ_VJSON(СписокПеременных)
GETVJSON(СписокПеременных)
Функция формирует текст в формате JSON по указанному списку переменных. В качестве параметра передается строка, содержащая имена заранее определенных переменных. Имена должны быть разделены пробелом или запятой. Полученная строка будет содержать объект json. Именами полей будут служить имена переменных. Значениями полей будут служить значения переменных. Если какой-то из перечисленных в списке переменных нет в вычислителе, то для поля с её названием подставляется значение null.

Параметры.
СписокПеременных - Строка со списком имен переменных.

Возвращаемое значение - Строка, содержащая требуемые данные в формате JSON.
Пример.

СФОРМИРОВАТЬ_VJSON("var1, var2 ").
Результат будет в виде json строки
{ "var1":ЗначениеПеременнойVar1, "var2":ЗначениеПеременнойVar2 }

РАЗОБРАТЬ_VJSON(СтрокаJson[,ИмяПеременной])
JSONTOVARS(СтрокаJson[,ИмяПеременной])
Функция создает переменные с именами равными именам полей json объекта первого уровня и значениями равными значениям полей этого объекта.
Например:
РАЗОБРАТЬ_VJSON({""var1":ЗначениеПеременной1, "var2":ЗначениеПеременной2, "var3":ЗначениеПеременной3}");

В результате будут созданы переменные var1, var2, var3 со значениями ЗначениеПеременной1, ЗначениеПеременной2 и ЗначениеПеременной3, соответственно. Если какая-то из переменных уже была определена, то её значение сменится на соответствующее ей значение поля в объекте. Для вложенных объектов и массивов создаются строковые переменные, содержащие тексты вложенных объектов. При разборе массивов создаются элементы массива с именем равным имени поля и значениями равными значениям элементов массива. Для служебного значения null будет создана переменная без значения. При разборе безымянного массива необходимо указать второй параметр функции с именем формируемого массива.
Параметры.
СтрокаJson- Строка json для разбора.
Возвращаемое значение - Число целое, равное количеству созданных переменных.

ДАННЫЕ_JSON(ИмяПоля, Значение)
JSONDATA(ИмяПоля, Значение)
Функция формирует текст в формате JSON для указанного имени поля и его значения.
Параметры.
ИмяПоля- Строка имя поля.
Значение- Выражение любого типа для представления в формате JSON.
Возвращаемое значение - Строка, содержащая требуемые данные в формате JSON.
Пример.
ДАННЫЕ_JSON("Название", "ООО ""Ромашка""")
Результат: "Название":"ООО \"Ромашка\"

ПОЛЕ_JSON(JSONТекст,ИмяПоля | ИндексПоля, ЗначениеПоУмолчанию)
JSONFIELD(JSONТекст,ИмяПоля | ИндексПоля, ЗначениеПоУмолчанию)
Функция извлекает значение поля из объекта JSON. Если указанного поля в объекте нет, то возвращается значение по умолчанию.
Параметры.
JSONТекст- Строка, текст в формате JSON, представляющий единичный объект.
ИмяПоля- Строка, имя поля, значение которого нужно получить.
ИндексПоля- Число целое, номер поля, начиная с нуля, значение которого нужно получить. Номер поля можно указывать только для объектов типа Массив, в котором нет названий полей.
ЗначениеПоУмолчанию- Выражение, значение которого возвращается в случае отсутствия поля в объекте.
Возвращаемое значение - Значение поля.

ЗАГРУЗИТЬДАННЫЕ(ИмяФайлаОписания, ИмяФайлаДанных)
LOADDATA(ИмяФайлаОписания, ИмяФайлаДанных)
Функция загружает в базу данных данные из файла ИмяФайлаДанных. В качестве правил для загрузки используется XML описание из файла ИмяФайлаОписания. Функция автоматически производит преобразования из кодировки UTF-16 и UTF-8 в ANSI. При этом, файл в кодировке UTF-16 должен иметь префикс 0xFEFF или 0xFFEF. Файл в кодировке UTF-8 должен иметь префикс 0xEFBBBF.
Параметры.
ИмяФайлаОписания - Строка, имя файла с XML описанием правил загрузки данных.

ИмяФайлаДанных - Строка, имя файла с данными для загрузки в XML формате.
Возвращаемое значение - Логическое. ИСТИНА, если загрузка произведена без ошибок, иначе ЛОЖЬ.

ВЫГРУЗИТЬДАННЫЕ(КодШаблона)
EXPORTDATA(КодШаблона)
Функция выгружает из базы данных данные в файл, согласно указанному шаблону экспорта данных.
Параметры.
КодШаблона - Строка, имя или код шаблона из справочника экспорта данных.
Возвращаемое значение - Логическое. Истина, если выгрузка произведена без ошибок, иначе Ложь.

ТИП(Выражение)
TYPE(Выражение)
Возвращает тип значения указанного выражения.
Параметры.
Выражение - Выражение, значение которого необходимо определить. Так как в системе отсутствует строгая типизация данных, для определения типа Выражения необходимо его вычислить, и по результату вычисления определить тип.
Возвращаемое значение - Строка. В зависимости от определенного типа Выражения, возвращается один из символов. Все символы заглавные латинские.
'U' - Выражение не может быть вычислено.
'I' - Тип результата вычисления Выражения Число целое.
'B' - Тип результата вычисления Выражения Число вещественное.
'C' - Тип результата вычисления Выражения Строка.
'L' - Тип результата вычисления Выражения Логический
'D' - Тип результата вычисления Выражения ДатаВремя.

ФУНКЦИИ ДЛЯ РАБОТЫ С КОНТЕКСТАМИ

ДОБАВИТЬКОНТЕКСТ(СтрокаSQL, ИмяКонтекста[, ТекстСообщения,[НомерСоединения | СписокФилиалов]])
ADDCONTEXT(СтрокаSQL, ИмяКонтекста[, ТекстСообщения,[НомерСоединения | СписокФилиалов]])
Выполняет указанный запрос SQL и формирует таблицу контекста из результата выполнения запроса для дальнейшего использования. Функция позволяет указать номер соединения или список филиалов для выборки данных. Добавленный контекст становится текущим контекстом. Если сервер вернул ошибку выполнения запроса, то функция возвращает Ложь и записывает полученное от сервера сообщение в журнал сообщений и ошибок (**errorlog**).
Параметры.
СтрокаSQL - Текст SQL запроса, из результата выполнения которого будет сформирован контекст. Если строка запроса начинается со слова **LOCAL:**, то это будет означать, что запрос не будет отправлен на SQL сервер, а будет создана локальная таблица с описанной структурой. Например, строка ***"LOCAL: field1 char, field2 int, field3 datetime, field4 float"*** будет означать, что запрос на сервер не будет отправлен, а будет создана локальная таблица с полями field1,…, field4 указанных типов.
ИмяКонтекста - Имя, которое будет присвоено полученному контексту.
ТекстСообщения - Текст сообщения, который будет выведен во время выполнения запроса. При продолжительном времени выполнения запросов позволяет известить пользователя о происходящем действии. Если параметр не указан, то выводится стандартное сообщение.
НомерСоединения - позволяет указать номер источника данных, из которого будет произведена выборка. Данный номер возвращается функцией ДобавитьСоединение.
СписокФилиалов - Список филиалов через пробел, из баз которых будет произведена выборка данных и объединена в единый контекст.

Возвращаемое значение - логическое. Истина, если запрос был выполнен удачно и контекст был сформирован. Ложь, если сервер вернул ошибку.

ДОБАВИТЬСТРОКИ(КоличествоСтрок[, ИмяКонтекста[, Заполнять[, Выражение[, СписокВключаемыхПолей[, СписокИсключаемыхПолей]]]]])
APPEND(КоличествоСтрок[, ИмяКонтекста[, Заполнять[, Выражение[, СписокВключаемыхПолей[, СписокИсключаемыхПолей]]]]])
Добавляет указанное количество строк в указанный контекст. Если параметр **Заполнять** имеет значение **ИСТИНА**, то новая запись будет заполнена значениями переменных, соответствующих именам полей
Параметры.
КоличествоСтрок - Число. Количество строк, которое необходимо добавить. За один вызов не может быть добавлено более 1000 строк.
ИмяКонтекста - Строка. Указывает, в какой контекст необходимо добавить строки. Если параметр не указан, строки будут добавлены в текущий контекст.
Заполнять - Логическое, если ИСТИНА, то новая запись будет заполнена значениями соответствующих переменных.
Выражение - Строка, если указано, то будет вычисляться перед добавлением каждой новой строки.
СписокВключаемыхПолей – Строка, если указано, то только перечисленные поля будут заполнены значениями.
СписокИсключаемыхПолей – Строка, если указано, то перечисленные поля не будут заполнены значениями.
Возвращаемое значение - функция не возвращает значений.

КОНЕЦКОНТЕКСТА([ИмяКонтекста])
ENDOFVIEW([ИмяКонтекста])
Функция возвращает ИСТИНУ, если в указанном контексте указатель строк находится за пределами последней строки.
Параметры.
ИмяКонтекста - Строка, имя контекста, для которого проводится проверка. Если параметр не указан, проверяется текущий контекст.
Возвращаемое значение - логическое. ИСТИНА, если достигнут конец контекста. Иначе - ЛОЖЬ.

ПЕРЕЙТИ(НомерСтроки [, ИмяКонтекста])
GO(НомерСтроки [, ИмяКонтекста])
Выполняет переход на указанную строку в указанном контексте. Если имя контекста не указано, используется текущий контекст. Если НомерСтроки меньше или равен нулю, то переход осуществляется на первую строку контекста. Если НомерСтроки больше или равен количеству строк в контексте, то переход осуществляется на последнюю строку контекста.
Параметры.
НомерСтроки - Число, номер строки, на которую необходимо произвести переход.
ИмяКонтекста - Строка. Указывает, в каком контексте необходимо осуществить переход. Если параметр не указан, переход будет осуществлен в текущем контексте.
Возвращаемое значение - нет.

ПЕРЕЙТИВКОНЕЦ([ИмяКонтекста])
GOBOTTOM([ИмяКонтекста])
Выполняет переход на последнюю строку в указанном контексте. Если имя контекста не указано, используется текущий контекст.
Параметры.
ИмяКонтекста - Строка. Указывает, в каком контексте необходимо осуществить переход. Если параметр не указан, переход будет осуществлен в текущем контексте.
Возвращаемое значение - нет.

ПЕРЕЙТИВНАЧАЛО([ИмяКонтекста])
GOTOP([ИмяКонтекста])
Выполняет переход на первую строку в указанном контексте. Если имя контекста не указано, используется текущий контекст.
Параметры.
ИмяКонтекста - Строка. Указывает, в каком контексте необходимо осуществить переход. Если параметр не указан, переход будет осуществлен в текущем контексте.
Возвращаемое значение - нет.

КОЛИЧЕСТВОСТРОК([ИмяКонтекста])
RECCOUNT([ИмяКонтекста])
Функция возвращает количество строк в указанном контексте.
Параметры.
ИмяКонтекста - Строка, имя контекста, количество строк которого определяется. Если параметр не указан, то возвращается количество строк в текущем контексте.
Возвращаемое значение - Число целое, равное количеству строк в указанном контексте.

УДАЛИТЬКОНТЕКСТ(ИмяКонтекста)
REMOVECONTEXT(ИмяКонтекста)
Функция удаляет из системы контекст с указанным именем и освобождает все связанные с контекстом ресурсы. После удаления контекст становится недоступным для использования. В случае удаления контекста, по которому производится вывод секции, вывод секции прекращается на следующей итерации и производится стандартный выход из секции.
Параметры.
ИмяКонтекста - Строка, имя контекста, который необходимо удалить.
Возвращаемое значение - нет.

ПРОПУСТИТЬ([КоличествоСтрок [, ИмяКонтекста]])
SKIP([КоличествоСтрок [, ИмяКонтекста]])
Выполняет переход на строку, отстоящую от текущей на указанное КоличествоСтрок в указанном контексте. Если имя контекста не указано, используется текущий контекст. Если КоличествоСтрок меньше нуля, то переход осуществляется к началу контекста, иначе к концу контекста.
Параметры.
КоличествоСтрок - Число, количество строк, которое необходимо пропустить. Если параметр не указан, то пропускается одна строка.
ИмяКонтекста - Строка. Указывает, в каком контексте необходимо осуществить переход. Если параметр не указан, переход будет осуществлен в текущем контексте.
Возвращаемое значение - нет.

ВЫБРАТЬКОНТЕКСТ(ИмяКонтекста)
SELECTCONTEXT(ИмяКонтекста)
Выбирает в качестве текущего контекста указанный контекст.
Параметры.
ИмяКонтекста - Строка. Указывает, какой контекст следует сделать текущим.
Возвращаемое значение - логическое. ИСТИНА, если указанный контекст существует и был выбран в качестве текущего. Иначе - ЛОЖЬ.

НОМЕРСТРОКИ([ИмяКонтекста])
RECNO([ИмяКонтекста])
Функция возвращает номер текущей строки в указанном контексте.
Параметры.

ИмяКонтекста - Строка, имя контекста, для которого определяется номер текущей строки. Если параметр не указан, то возвращается номер строки в текущем контексте.
Возвращаемое значение - Число целое, равное номеру текущей строки в указанном контексте.

СОЗДАТЬИНДЕКС(ИмяКонтекста, Выражение[, ДлинаКлюча[, Флаги[, ИмяФайла]]])
CREATEINDEX(ИмяКонтекста, Выражение[, ДлинаКлюча[, Флаги[, ИмяФайла]]])
Функция создает файл индекса с уникальным именем для указанного контекста.Параметр Выражение должен содержать индексное выражение.
Параметры.
ИмяКонтекста- Строка, имя существующего контекста, добавленного функцией **ДОБАВИТЬКОНТЕКСТ()**.
Выражение - Строка содержащая имена полей, допустимых функций и их комбинации. Значение выражения будет являться ключом индекса.
ДлинаКлюча - Число целое, указывает размер ключа для текстового ключа. Если параметр не указан, то используется длина значения переданного выражения.
Флаги - Число целое, 1 - Создавать уникальный индекс, 0 - создавать не уникальный индекс. Если параметр не указан, то используется значение 0.
ИмяФайла - Строка, имя файла для создаваемого индекса. Если параметр не указан, то будет использовано имя временного файла. Возвращаемое значение - число целое. Больше или равно 0 - если файл создан успешно. Это значение должно использоваться во всех функциях работы с индексами. Иначе код ошибки.
Пример. СоздатьИндекс('Список', 'Наименование')

НАЙТИ(ИмяКонтекста, Выражение[| ИмяПоля, Значение]),
FIND(ИмяКонтекста, Выражение[| ИмяПоля, Значение])
Функция осуществляет поиск записи в указанном контексте, для которой значение указанного выражения будет равно ИСТИНЕ. Вычисление Выражения должно возвращать логический результат.
Параметры.
ИмяКонтекста - Строка, имя существующего контекста, добавленного функцией **ДОБАВИТЬКОНТЕКСТ()**.
Выражение - Строка, содержащая выражение значение которого будет вычислено для всех записей контекста, пока не будет найдена строка, для которой значение выражения будет равно ИСТИНЕ.
ИмяПоля - Строка, имя поля контекста, в котором будет осуществлен поиск указанного значения.
Значение - Значение любого типа, поиск которого будет осуществлен.
Комментарий. Следует обратить внимание на то, что поиск по выражению происходит с учетом текущего открытого индексного файла. Поэтому изменение значения ключевого поля может изменить работу функции **ПРОДОЛЖИТЬПОИСК()**. Поиск по значению поля производится без учета текущего открытого индексного файла, поэтому изменение значения ключевого поля не влияет на дальнейшую работу функции **ПРОДОЛЖИТЬПОИСК()**.
Возвращаемое значение - логическое. **ИСТИНА**, если запись найдена. Иначе **ЛОЖЬ**.

НАЙТИПОИНДЕКСУ(ИмяКонтекста, ЗначениеКлюча[, НомерИндекса])
FINDINDEX(ИмяКонтекста, ЗначениеКлюча[, НомерИндекса])
Функция осуществляет поиск указанного значения ключа в индексе. Если такой ключ есть в индексе, то номер записи в контексте устанавливается на найденную запись. Если не указан номер индекса для поиска, то используется последний созданный индекс, или индекс указанный функцией **ВЫБРАТЬИНДЕКС()**.
Параметры.
ИмяКонтекста - Строка, имя существующего контекста, добавленного функцией **ДОБАВИТЬКОНТЕКСТ()**.
ЗначениеКлюча - Значение ключа, которое будет найдено в индексе.
НомерИндекса - Число целое, номер индекса, который вернула функция **СОЗДАТЬИНДЕКС()**.

Возвращаемое значение – логическое. **ИСТИНА**, если запись найдена. Иначе – **ЛОЖЬ**.

ПРОДОЛЖИТЬПОИСК(ИмяКонтекста)
FINDCONTINUE(ИмяКонтекста)
Функция осуществляет поиск следующей удовлетворяющей условиям поиска записи, начиная со следующей записи. Если последний поиск осуществлялся функцией **НАЙТИ**(), то будет продолжен обход контекста и поиск следующей записи удовлетворяющей Выражению поиска. В случае поиска по индексу, будет найдена следующая запись с таким же ключом.
Параметры.
ИмяКонтекста – Строка, имя существующего контекста, добавленного функцией **ДОБАВИТЬКОНТЕКСТ**().

Возвращаемое значение – логическое. **ИСТИНА**, если поиск успешен. Иначе – **ЛОЖЬ**.

ИЗМЕНИТЬПОЛЕ(ИмяКонтекста, ИмяПоля | НомерПоля, ЗначениеПоля)
REPLACE(ИмяКонтекста, ИмяПоля | НомерПоля, ЗначениеПоля)
Функция изменяет значение указанного поля на указанное значение. Если на момент вызова функции у контекста открыты индексы, то будет произведено изменение ключей. Если открыты уникальные индексы и новое значение ключа будет неуникальным, то функция вернет **ЛОЖЬ** и значение поля не будет изменено.
Параметры.
ИмяКонтекста – Строка, имя существующего контекста, добавленного функцией **ДОБАВИТЬКОНТЕКСТ**().
НомерПоля – Число целое. Номер поля в контексте, значение которого надо изменить. Нумерация полей начинается с 1.
ИмяПоля – Строка, имя поля, значение которого надо изменить.
Значение – Значение любого типа, которое будет установлено для указанного поля.

Возвращаемое значение – логическое. **ИСТИНА**, если значение поля успешно изменено. Иначе – **ЛОЖЬ**.
ЗНАЧЕНИЕПОЛЯ(ИмяКонтекста, ИмяПоля | НомерПоля[,ЗначениеПоУмолчанию])
FIELDVALUE(ИмяКонтекста, ИмяПоля | НомерПоля[,ЗначениеПоУмолчанию])
Функция возвращает значение поля.
Параметры.
ИмяКонтекста – Строка, имя существующего контекста, добавленного функцией **ДОБАВИТЬКОНТЕКСТ()**.
НомерПоля – Число целое. Номер поля в контексте, значение которого надо получить. Нумерация полей начинается с 1.
ИмяПоля – Строка, имя поля, значение которого надо получить.
Если указан только один параметр, то он интерпретируется как имя/номер поля в текущем контексте.
ЗначениеПоУмолчанию – Значение, возвращаемое в случае отсутствия указанного поля.
Возвращаемое значение – зависит от типа поля. Если указанного поля нет в контексте, то возвращается значение по умолчанию.

ТИПЗНАЧЕНИЯПОЛЯ(ИмяКонтекста, ИмяПоля | НомерПоля)
FIELDVALUETYPE(ИмяКонтекста, ИмяПоля | НомерПоля
Функция возвращает тип значения поля контекста.
Параметры.
ИмяКонтекста – Строка, имя существующего контекста, добавленного функцией **ДОБАВИТЬКОНТЕКСТ()**.
НомерПоля – Число целое. Номер поля в контексте, значение которого надо получить. Нумерация полей начинается с 1.
ИмяПоля – Строка, имя поля, значение которого надо получить.

Если указан только один параметр, то он интерпретируется как имя/номер поля в текущем контексте.
Возвращаемое значение - строка, состоящая из одного символа означающего тип поля данных. Если указанного поля нет в контексте, то возвращается **'U'**.

ВЫБРАТЬИНДЕКС(ИмяКонтекста, НомерИндекса)
SELECTINDEX(ИмяКонтекста, НомерИндекса)
Функция устанавливает текущий активный индекс. Выбранный индекс будет использоваться при перемещениях по контексту функциями **ПЕРЕЙТИВНАЧАЛО**(), **ПЕРЕЙТИВКОНЕЦ**(), **ПРОПУСТИТЬ**(), а также при поиске функцией **НАЙТИПОИНДЕКСУ**() без указания номера индекса.
Параметры.
ИмяКонтекста - Строка, имя существующего контекста, добавленного функцией **ДОБАВИТЬКОНТЕКСТ**().
НомерИндекса - Число целое, номер индекса, который вернула функция **СОЗДАТЬИНДЕКС**().

Возвращаемое значение - логическое. **ИСТИНА**, если индекс успешно выбран. Иначе **ЛОЖЬ**.

ЗАГРУЗИТЬ(ИмяКонтекста, ТипИсточникаДанных, ИсточникДанных[, Параметр1[, Параметр2[, ЗагружаемыеБлокиДанных[, ПропускаемыеБлокиДанных]]]])
UPLOAD(ИмяКонтекста, ТипИсточникаДанных, ИсточникДанных[, Параметр1[, Параметр2[, ЗагружаемыеБлокиДанных[, ПропускаемыеБлокиДанных]]]])
Функция загружает данные в контекст из указанного источника. Параметр ТипИсточникаДанных определяет каким образом необходимо получить данные из источника данных.
Параметры.
ИмяКонтекста - Строка, имя существующего контекста, добавленного функцией ДОБАВИТЬКОНТЕКСТ().
ТипИсточникаДанных - Строка, содержащая тип источника данных. Допустимые значения для данного параметра:
- **CSV** - текстовый файл с разделителями полей,
- **XML** - текстовый файл в формате XML,
- **XMLSTRING** - строка в формате XML,
- **JSON** - строка в формате JSON.

ИсточникДанных - Строка содержащая имя файла для загрузки данных.
Параметр1 - Строка содержащая дополнительные параметры. Для типа **CSV** может быть указан разделитель полей. Для типа **XML** необходимо указать тэг разделитель строк. Для типа **JSON** может быть скрипт, выполняемый при добавлении новой строки. Он может использоваться, например, для инициализации значений полей, которых может не быть в данных.
Параметр2 - Строка содержащая дополнительные параметры. Для всех типов может быть указано выражение для преобразования значения поля. При вычислении значения выражения могут быть использованы дополнительные переменные:
- **ИмяПоля** - Строка, имя загружаемого поля,
- **НомерПоля** - Число целое, номер загружаемого поля,
- **ЗначениеПоля**- Строка, загружаемое значение.

ЗагружаемыеБлокиДанных - Строка содержащая список блоков данных (тэгов), которые необходимо загрузить. Остальные блоки будут пропущены. Если параметр не указан или пустой, то игнорируется и загружаются все блоки данных. Используется для типов источника данных **XML**, **XMLSTRING**, **JSON**.
ПропускаемыеБлокиДанных - Строка содержащая список блоков данных (тэгов), которые необходимо исключить из загрузки. Остальные блоки будут

загружены. Если параметр не указан или пустой, то игнорируется. Используется для типов источника данных **XML, XMLSTRING, JSON.**

Возвращаемое значение – логическое. Истина, если операция прошла без ошибок.

ВЫГРУЗИТЬ(ИмяТаблицы[, СписокПолей[, СписокИсключений[, ИмяКонтекста[, НомерСоединения]]]])
DOWNLOAD(ИмяТаблицы[, СписокПолей[, СписокИсключений[, ИмяКонтекста[, НомерСоединения]]]])

Функция выгружает данные из контекста в таблицу базы данных. Таблица получатель данных должна содержать в себе список необходимых полей, типы данных которых, должны соответствовать типам данных полей контекста.

Параметры.

ИмяТаблицы - Строка, имя таблицы в базе данных. Если таблица не существует, то будет предпринята попытка создать таблицу по структуре выгружаемых полей. Если в качестве параметра передана пустая строка, то будет сгенерировано новое уникальное имя таблицы. Это имя будет возвращено функцией, при успешном завершении.

СписокПолей - Строка, содержащая список полей, разделенных пробелом или запятой. Если список указан, и он не пустой, то будут выгружены только поля из этого списка.

СписокИсключений - Строка, содержащая список полей, разделенных пробелом или запятой. Если список указан, и он не пустой, то поля из этого списка не будут выгружаться.

ИмяКонтекста - Строка, имя существующего контекста, добавленного функцией ДОБАВИТЬКОНТЕКСТ( ).

НомерСоединения - Число целое. Позволяет указать номер источника данных, в котором расположена заполняемая таблица. Этот номер возвращается функцией ДОБАВИТЬСОЕДИНЕНИЕ( ).

Возвращаемое значение - логическое или строка. Если имя было указано, то логическое значение ИСТИНА укажет на удачное завершение, иначе будет возвращено значение ЛОЖЬ. Если имя таблицы не было указано, то будет возвращена строка с именем новой таблицы. Если имя таблицы не было указано и функция завершилась неудачно, то будет возвращена пустая строка.

ЗАГРУЗИТЬJSON(ИмяКонтекста, СтрокаJSON[,ИмяПоля])
UPLOADJSON(ИмяКонтекста, СтрокаJSON[,ИмяПоля])

Функция загружает данные в контекст из указанной строки JSON. Если строка JSON – это массив, то формируется контекст со строками, содержащими элементы массива. Иначе, в контекст загружается текущий объект JSON. Если указанный контекст существует, то он будет удален и создан заново со структурой загружаемого объекта. Все загружаемые поля имеют текстовый тип.

Параметры.

ИмяКонтекста– Строка, имя контекста, в который будет загружен объект.

СтрокаJSON– Строка, содержащая JSON объект для загрузки.

ИмяПоля– Строка, имя поля для объектов без имени. Если таких полей несколько в объекте, то к имени поля будет прибавляться суффикс из номера поля без имени.

Возвращаемое значение - логическое. Истина, если операция прошла без ошибок.

ИМЯКОНТЕКСТА( )
CONTEXTNAME( )

Функция возвращает имя текущего выбранного контекста. Если ни один контекст не выбран, то функция вернет пустое значение.

Параметры.

нет.

Возвращаемое значение - Строка, имя текущего выбранного контекста, или пустая строка.

ЧИТАТЬЗАПИСЬ([ИмяКонтекста[, СписокВключаемыхПолей[, СписокИсключаемыхПолей]]])
LOADRECORD([ИмяКонтекста[, СписокВключаемыхПолей[, СписокИсключаемыхПолей]]])
Функция загружает значения полей указанного контекста в переменные с именами соответствующими именам полей. Если на момент вызова функции уже определены переменные с требуемыми именами, то значения таких переменных становятся равными значениям полей в текущей записи контекста. Если таких переменных еще не было определено, то они создаются и им присваиваются соответствующие значения. Если имя контекста не указано, то функция загружает поля текущего контекста.
Параметры.
ИмяКонтекста - Строка, имя контекста, поля которого необходимо загрузить.
СписокВключаемыхПолей – Строка, если указано, то только перечисленные поля будут заполнены значениями.
СписокИсключаемыхПолей – Строка, если указано, то перечисленные поля не будут заполнены значениями.
Возвращаемое значение - логическое, **ИСТИНА**, если значения загружены успешно, иначе **ЛОЖЬ**.

ЗАГРУЗИТЬЗАПИСЬ([ИмяКонтекста[, СписокВключаемыхПолей[, СписокИсключаемыхПолей]]])
GATHERRECORD([ИмяКонтекста[, СписокВключаемыхПолей[, СписокИсключаемыхПолей]]])
Функция загружает значения переменных в поля указанного контекста с именами соответствующими именам переменных. Если на момент вызова функции уже определены переменные с именами, такими же как имена полей, то значения таких переменных переносятся в значения полей в текущей записи контекста. Если таких переменных еще не было определено, то значения полей не изменяются. Если имя контекста не указано, то функция устанавливает значения полей текущего контекста.
Параметры.
ИмяКонтекста - Строка, имя контекста, поля которого необходимо установить.
СписокВключаемыхПолей – Строка, если указано, то только перечисленные поля будут заполнены значениями.
СписокИсключаемыхПолей – Строка, если указано, то перечисленные поля не будут заполнены значениями.
Возвращаемое значение - логическое, **ИСТИНА**, если значения установлены успешно, иначе **ЛОЖЬ**.

ФУНКЦИИ ДЛЯ РАБОТЫ С ТАБЛИЦАМИ

ТАБЛИЦАСОЗДАТЬ(ИмяФайлаТаблицы, СтрокаОписанияСтруктуры[, Флаги[, Версия]])
TABLECREATE(ИмяФайлаТаблицы, СтрокаОписанияСтруктуры[, Флаги[, Версия]])
Функция создает файл с указанным именем в формате DBF. Описание структуры должно содержать имена полей и их типы. Имя файла может содержать переменную %TEMP%, которая будет заменена на путь к каталогу временных файлов.
Параметры.
ИмяФайлаТаблицы - Строка содержащая полное имя (с путем) создаваемого файла.
ОписаниеСтруктуры - Строка содержащая описание структуры создаваемого файла. Формат строки: ИмяПоля1 БуквaТипаДанных[(Размер, ЗнаковПослеЗапятой)][,...], где ИмяПоля - строка состоящая из латинских букв и цифр, начинающаяся с буквы.
БуквaТипаДанных - латинская буква соответствующая типу данных:
C[(КоличествоСимволов)] - символьное поле указанного размера,
N[(ВсегоЗнаков[, ПослеЗапятой])] - Число с фиксированной десятичной точкой,
L - логическое,
D - дата (без времени),
M - мемо поле (символьное без ограничения длины),
I - целочисленное,
F - вещественное.
Флаги - Число целое,1 - Не перезаписывать имеющийся файл, 0 - Перезаписывать имеющийся файл с указанным именем. Если параметр не указан, то используется значение 0.

Версия - Число целое, 3 - Создать файл в формате DBase III+, 4 - Создать файл в формате DBase IV. Если параметр не указан, то используется значение 3.

Возвращаемое значение - число целое. Больше или равное 0 - если файл создан успешно. Это значение должно использоваться во всех функциях работы с таблицей. Иначе код ошибки.

Пример. ТаблицаСоздать('%TEMP%Таблица1', 'field1 C(50), field2 N(10, 3), field3 M, field4 D, field5 I, field6 F')

ТАБЛИЦАСОЗДАТЬИНДЕКС(НомерТаблицы, Выражение[, ИмяФайлаИндекса[, Флаги]])
TABLECREATEINDEX(НомерТаблицы, Выражение[, ИмяФайлаИндекса[, Флаги]])
Функция создает файл индекса с указанным именем для указанной таблицы. Параметр Выражение должен содержать индексное выражение.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
Выражение - Строка содержащая имена полей, допустимых функций и их комбинации. Значение выражения будет являться ключом индекса.
ИмяФайлаИндекса - Строка содержащая полное имя (с путем) создаваемого файла. Если имя файла не указано, то файл будет создан в каталоге для временных файлов с уникальным именем.
Флаги - Число целое, 1 - Создавать уникальный индекс, 0 - создавать не уникальный индекс. Если параметр не указан, то используется значение 0.
Возвращаемое значение - число целое. Больше или равно 0 - если файл создан успешно. Это значение должно использоваться во всех функциях работы с индексами. Иначе код ошибки.
Пример. **ТаблицаСоздатьИндекс**(0, 'Индекс1', 'field1')

ТАБЛИЦАОТКРЫТЬ(ИмяФайлаТаблицы)
TABLEOPEN(ИмяФайлаТаблицы)
Функция открывает существующий файл с указанным именем в формате DBF.
Параметры.
ИмяФайлаТаблицы - Строка содержащая полное имя (с путем) создаваемого файла.
Возвращаемое значение - число целое. Больше или равное 0 - если файл открыт успешно. Это значение должно использоваться во всех функциях работы с таблицей. Иначе, код ошибки. Пример. **ТаблицаОткрыть**('Таблица1')

ТАБЛИЦАОТКРЫТЬИНДЕКС(НомерТаблицы, ИмяФайлаИндекса)
TABLEOPENINDEX(НомерТаблицы, ИмяФайлаИндекса)
Функция открывает существующий файл индекса для указанной таблицы. В связи с тем, что в формате индексного файла не предусмотрено хранение имени исходной таблицы, то никаких проверок соответствия открываемого файла файлу таблицы не производится. Открываемый файл добавляется в список открытых индексов таблицы и может быть выбран функцией **ТАБЛИЦАВЫБРАТЬИНДЕКС**( ). Все изменения производимые с таблицей отражаются в открытых индексах автоматически. Если файл индекса не открыт, то изменения в таблице данных в нем не отражаются.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
ИмяФайлаИндекса - Строка содержащая полное имя (с путем) открываемого файла.

Возвращаемое значение - число целое. Больше или равное 0 - если файл открыт успешно. Это значение должно использоваться во всех функциях работы с индексами. Иначе код ошибки.

Пример. ТаблицаОткрытьИндекс('Таблица1')

ТАБЛИЦАЗАКРЫТЬ(НомерТаблицы[, Флаг])
TABLECLOSE(НомерТаблицы[, Флаг])
Функция закрывает указанную таблицу, все связанные открытые индексы и файлы с мемо полями. Также, освобождаются все занятые таблицей ресурсы. Если указан параметр Флаг, то будут удалены все связанные с таблицей и индексами файлы.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
Флаг - Число целое,1 - Удалять файлы таблицы, 0 - не удалять файлы таблицы. Если параметр не указан, то используется значение 0.
Возвращаемое значение - логическое. **ИСТИНА**, если операция завершена успешно. Иначе - **ЛОЖЬ**.

ТАБЛИЦАЗАКРЫТЬИНДЕКС(НомерИндекса[, Флаг])
TABLECLOSEINDEX(НомерИндекса[, Флаг])
Функция закрывает указанный индекс и освобождает все занятые индексом ресурсы.
Параметры.
НомерИндекса - Число целое, номер индекса, который вернула функция **ТАБЛИЦАСОЗДАТЬИНДЕКС**() или **ТАБЛИЦАОТКРЫТЬИНДЕКС**().
Флаг - Число целое, 1 - Удалять файл индекса, 0 - не удалять файл индекса. Если параметр не указан, то используется значение 0.
Возвращаемое значение - логическое. **ИСТИНА**, если операция завершена успешно. Иначе **ЛОЖЬ**.

ТАБЛИЦАПЕРЕЙТИ(НомерТаблицы, НомерЗаписи)
TABLEGOTO(НомерТаблицы, НомерЗаписи)
Функция позиционирует указатель записи для указанной таблицы на указанной записи.
Параметры.

НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
НомерЗаписи - Число целое, номер записи на которую надо перейти.

Возвращаемое значение - логическое. **ИСТИНА**, если операция завершена успешно. Иначе - **ЛОЖЬ**.

ТАБЛИЦАПЕРЕЙТИВНАЧАЛО(НомерТаблицы)
TABLEGOTOP(НомерТаблицы)
Функция позиционирует указатель записи для указанной таблицы на первой записи. Если в момент вызова функции у таблицы открыт индекс, то позиционирование производится на первой записи согласно сортировке в индексе.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
Возвращаемое значение - логическое. **ИСТИНА**, если операция завершена успешно. Иначе - **ЛОЖЬ**.

ТАБЛИЦАПЕРЕЙТИВКОНЕЦ(НомерТаблицы)
TABLEGOBOTTOM(НомерТаблицы)
Функция позиционирует указатель записи для указанной таблицы на последней записи. Если в момент вызова функции у таблицы открыт индекс, то позиционирование производится на последней записи согласно сортировке в индексе.
Параметры.
НомерТаблицы- Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
Возвращаемое значение - логическое. **ИСТИНА**, если операция завершена успешно. Иначе - **ЛОЖЬ**.

ТАБЛИЦАПРОПУСТИТЬ(НомерТаблицы[, КоличествоЗаписей= 1])
TABLESKIP(НомерТаблицы[, КоличествоЗаписей= 1])
Функция позиционирует указатель записей в указанной таблице на записи, отстоящей от текущей на указанное количество записей. Если параметр КоличествоЗаписей равен нулю, то переход не осуществляется. Если больше нуля, то переход осуществляется в направлении конца таблицы. Если меньше нуля, то в направлении к началу таблицы.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
КоличествоЗаписей - Число целое, количество записей которое надо пропустить. Если параметр не указан, то принимается значение 1.

Возвращаемое значение - логическое. **ИСТИНА**, если операция завершена успешно. Иначе - **ЛОЖЬ**.

ТАБЛИЦАКОНЕЦФАЙЛА(НомерТаблицы)
TABLEEOF(НомерТаблицы)
Функция проверяет, не находится ли указатель записей указанной таблицы дальше последней записи.
Параметры.
НомерТаблицы- Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().

Возвращаемое значение - логическое. **ИСТИНА**, если указатель находится за пределами таблицы. Иначе - **ЛОЖЬ**.

ТАБЛИЦАДОБАВИТЬЗАПИСЬ(НомерТаблицы[, СписокПолей])
TABLEAPPEND(НомерТаблицы[, СписокПолей])
Функция добавляет запись в указанную таблицу и устанавливает значения полей, которые указаны в списке. Если список полей не указан и перед вызовом данной функции была вызвана функция ТАБЛИЦАПРИВЯЗАТЬПОЛЯ( ), то будут установлены значения ранее указанных полей. Если на момент вызова функции у таблицы открыты индексы, то будет произведено добавление ключей. Если открыты уникальные индексы и значение ключа для новой записи будет неуникальным, то функция вернет ЛОЖЬ и запись не будет добавлена.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
СписокПолей - Строка, список полей через разделитель (пробел или запятая), значения которых надо установить. Значения для полей будут взяты из значений переменных, имена которых соответствуют именам полей.

Возвращаемое значение - логическое. **ИСТИНА**, если запись успешно добавлена. Иначе - **ЛОЖЬ**.

ТАБЛИЦАИЗМЕНИТЬПОЛЕ(НомерТаблицы, ИмяПоля | НомерПоля, ЗначениеПоля)
TABLEREPLACE(НомерТаблицы, ИмяПоля | НомерПоля, ЗначениеПоля)
Функция изменяет значение указанного поля на указанное значение. Если на момент вызова функции у таблицы открыты индексы, то будет произведено изменение ключей. Если открыты уникальные индексы и новое значение ключа будет неуникальным, то функция вернет **ЛОЖЬ** и значение поля не будет изменено.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
НомерПоля - Число целое. Номер поля в таблице, значение которого надо изменить. Нумерация полей начинается с 1.
ИмяПоля - Строка, имя поля, значение которого надо изменить.

Значение - Значение любого типа, которое будет установлено для указанного поля.

Возвращаемое значение - логическое. **ИСТИНА**, если значение поля успешно изменено. Иначе **ЛОЖЬ**.

ТАБЛИЦАОБНОВИТЬЗАПИСЬ(НомерТаблицы)
TABLEUPDATE(НомерТаблицы)
Функция обновляет значения полей, которые были указаны при вызове функции **ТАБЛИЦАПРИВЯЗАТЬПОЛЯ**(), на значения соответствующих переменных. Если на момент вызова функции у таблицы открыты индексы, то будет произведено изменение ключей. Если открыты **уникальные** индексы и новое значение ключа будет неуникальным, то функция вернет ЛОЖЬ и изменение значений полей не будет произведено.
Параметры.
НомерТаблицы- Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**()или **ТАБЛИЦАОТКРЫТЬ**().
Возвращаемое значение - логическое. **ИСТИНА**, если запись успешно изменена. Иначе - **ЛОЖЬ**.

ТАБЛИЦАУДАЛИТЬЗАПИСЬ(НомерТаблицы[, НомерЗаписи | Выражение])
TABLEDELETE(НомерТаблицы[, НомерЗаписи | Выражение])
Функция помечает на удаление указанные записи в таблице. Если второй параметр не указан, то будет помечена текущая запись. Если указан номер записи, то будет помечена на удаление указанная запись. Если указана строка, содержащая выражение, то будут помечены те записи, значение выражения для которых равно ИСТИНЕ. В Выражении могут участвовать только те поля, которые были привязаны функцией **ТАБЛИЦАПРИВЯЗАТЬПОЛЯ**().
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
НомерЗаписи - Число целое. Номер записи, которую надо пометить на удаление.
Выражение - Строка, содержащая выражение значение которого будет вычислено для всех записей таблицы. Те записи, для которых значение выражения будет равно **ИСТИНЕ** будут помечены на удаление.
Возвращаемое значение - логическое. **ИСТИНА**, если запись(и) успешно помечена на удаление. Иначе **ЛОЖЬ**.

ТАБЛИЦАНАЙТИ(НомерТаблицы, Выражение)
TABLEFIND(НомерТаблицы, Выражение)
Функция осуществляет поиск записи, для которой значение указанного выражения будет равно **ИСТИНЕ**. В **Выражении** могут участвовать только те поля, которые были привязаны функцией **ТАБЛИЦАПРИВЯЗАТЬПОЛЯ**().
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
Выражение - Строка, содержащая выражение значение которого будет вычислено для всех записей таблицы, пока не будет найдена строка, для которой значение выражения будет равно **ИСТИНЕ**.

Возвращаемое значение - логическое. **ИСТИНА**, если запись найдена. Иначе **ЛОЖЬ**.

ТАБЛИЦАНАЙТИПОИНДЕКСУ(НомерТаблицы, ЗначениеКлюча[, НомерИндекса])
TABLEFINDINDEX(НомерТаблицы, ЗначениеКлюча[, НомерИндекса])
Функция осуществляет поиск указанного значения ключа в индексе. Если такой ключ есть в индексе, то номер записи в таблице устанавливается на найденную запись. Если не указан номер индекса для поиска, то используется последний открытый/созданный индекс, или индекс указанный функцией **ТАБЛИЦАВЫБРАТЬИНДЕКС**().
Параметры.

НомерТаблицы – Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
ЗначениеКлюча – Значение ключа, которое будет найдено в индексе.
НомерИндекса – Число целое, номер индекса, который вернула функция **ТАБЛИЦАСОЗДАТЬИНДЕКС**() или **ТАБЛИЦАОТКРЫТЬИНДЕКС**().
Возвращаемое значение – логическое. **ИСТИНА**, если запись найдена. Иначе – **ЛОЖЬ**.

ТАБЛИЦАПРОДОЛЖИТЬПОИСК(НомерТаблицы)
TABLEFINDCONTINUE(НомерТаблицы)
Функция осуществляет поиск следующей удовлетворяющей условиям поиска записи, начиная со следующей записи. Если последний поиск осуществлялся функцией **ТАБЛИЦАНАЙТИ**(), то будет продолжен обход таблицы и поиск следующей записи удовлетворяющей Выражению поиска. В случае поиска по индексу, будет найдена следующая запись с таким же ключом.
Параметры.
НомерТаблицы – Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().

Возвращаемое значение – логическое. **ИСТИНА**, если очередная запись найдена. Иначе – **ЛОЖЬ**.

ТАБЛИЦАВЫБРАТЬИНДЕКС(НомерТаблицы, НомерИндекса)
TABLEINDEX(НомерТаблицы, НомерИндекса)
Функция устанавливает текущий активный индекс. Выбранный индекс будет использоваться при перемещениях по таблице функциями **ТАБЛИЦАПЕРЕЙТИВНАЧАЛО**(), **ТАБЛИЦАПЕРЕЙТИВКОНЕЦ**(), **ТАБЛИЦАПРОПУСТИТЬ**(), а также при поиске функцией **ТАБЛИЦАНАЙТИПОИНДЕКСУ**() без указания номера индекса.
Параметры.
НомерТаблицы – Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
НомерИндекса – Число целое, номер индекса, который вернула функция **ТАБЛИЦАСОЗДАТЬИНДЕКС**() или **ТАБЛИЦАОТКРЫТЬИНДЕКС**().

Возвращаемое значение – логическое. **ИСТИНА**, если индекс успешно выбран. Иначе – **ЛОЖЬ**.

ТАБЛИЦАКОЛИЧЕСТВОЗАПИСЕЙ(НомерТаблицы)
TABLERECCOUNT(НомерТаблицы)
Функция возвращает физическое количество записей в таблице. Полученное значение включает в себя записи помеченные на удаление.
Параметры.
НомерТаблицы – Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
Возвращаемое значение – число целое равное количеству записей в таблице.

ТАБЛИЦАНОМЕРЗАПИСИ(НомерТаблицы)
TABLERECNO(НомерТаблицы)
Функция возвращает номер текущей записи в таблице.
Параметры.
НомерТаблицы – Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
Возвращаемое значение – число целое равное номеру текущей записи в таблице.

ТАБЛИЦАЗНАЧЕНИЕПОЛЯ(НомерТаблицы, НомерПоля | ИмяПоля)
TABLEFIELD(НомерТаблицы, НомерПоля | ИмяПоля)
Функция возвращает значение указанного поля в текущей строке таблицы.
Параметры.
НомерТаблицы – Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().

НомерПоля - Число целое, номер поля, значение которого необходимо получить.
ИмяПоля - Строка, имя поля, значение которого необходимо получить.
Возвращаемое значение - равно значению указанного поля в текущей строке таблицы. Если указатель записей находится за пределами строк таблицы, то возвращается пустое значение соответствующее типу поля. Если поля с указанным номером или именем нет в таблице, то возвращается **ЛОЖЬ**.

ТАБЛИЦАЧИТАТЬЗАПИСЬ(НомерТаблицы)
TABLELOADRECORD(НомерТаблицы)
Функция загружает значения полей таблицы в переменные с именами соответствующими именам полей. Если на момент вызова функции уже определены переменные с требуемыми именами, то значения таких переменных становятся равными значениям полей в текущей записи таблицы. Если таких переменных еще не было определено, то они создаются и им присваиваются соответствующие значения.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().

Возвращаемое значение - логическое, **ИСТИНА**, если значения загружены успешно, иначе **ЛОЖЬ**.

ТАБЛИЦАПРИВЯЗАТЬПОЛЯ(НомерТаблицы, СписокПолей)
TABLEBIND(НомерТаблицы, СписокПолей)
Функция привязывает указанные поля таблицы к переменным таким образом, что при любых перемещениях по записям таблицы значения переменных становятся равными значениям полей. При изменении значений связанных переменных значения полей автоматически изменяться не будут. Для обновления значений полей необходимо вызвать функцию **ТАБЛИЦАОБНОВИТЬЗАПИСЬ**(). Каждый последующий вызов функции **ТАБЛИЦАПРИВЯЗАТЬПОЛЯ**() для одной таблицы отменяет сделанные ранее привязки для этой таблицы. Таким образом, для отмены всех привязок необходимо вызвать функцию **ТАБЛИЦАПРИВЯЗАТЬПОЛЯ**() с пустым списком полей.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().
СписокПолей - Строка, список привязываемых полей разделенных запятыми. Если в качестве списка полей передана *, то будут привязаны все поля таблицы.
Возвращаемое значение - логическое, **ИСТИНА**, если привязки созданы успешно. Иначе - **ЛОЖЬ**.

ТАБЛИЦАУДАЛИТЬФАЙЛ(НомерТаблицы)
TABLEDROP(НомерТаблицы)
Функция закрывает указанную таблицу, после чего удаляет все связанные файлы: файл таблицы, файл с мемо полями, все открытые на момент вызова функции индексные файлы связанные с удаляемой таблицей.
Параметры.
НомерТаблицы - Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ**() или **ТАБЛИЦАОТКРЫТЬ**().

Возвращаемое значение - логическое, **ИСТИНА**, если операция прошла успешно. Иначе **ЛОЖЬ**.

ТАБЛИЦАЗАГРУЗИТЬ(НомерТаблицы, ТипИсточникаДанных, ИсточникДанных[, Параметр1[, Параметр2]])
TABLEUPLOAD(НомерТаблицы, ТипИсточникаДанных, ИсточникДанных[, Параметр1[, Параметр2]])
Функция загружает данные в таблицу из указанного источника. Параметр ТипИсточникаДанных определяет каким образом необходимо получить данные из источника.

Параметры.

НомерТаблицы – Число целое, номер таблицы, который вернула функция **ТАБЛИЦАСОЗДАТЬ()** или **ТАБЛИЦАОТКРЫТЬ()**.

ТипИсточникаДанных – Строка, содержащая тип источника данных. Допустимые значения для данного параметра:

- **CSV** – текстовый файл с разделителями полей,
- **DBF** – таблица в формате DBase III+ или IV,
- **XML** – текстовый файл в формате **XML**,
- **SQL** – загрузка данных из результата запроса **SQL**.

ИсточникДанных – Строка содержащая имя файла или строку запроса для загрузки данных.

Параметр1 – Строка содержащая дополнительные параметры. Для типа **CSV** может быть указан разделитель полей. Для типа **XML** необходимо указать тег-разделитель строк. Для типа **SQL** может быть указан номер соединения, который возвращает функция **ДобавитьСоединение( )**.

Параметр2 – Строка содержащая дополнительные параметры. Для всех типов может быть указано выражение для преобразования значения поля. При вычислении значения выражения могут быть использованы дополнительные переменные:

- **ИмяПоля** – Строка, имя загружаемого поля,
- **НомерПоля** – Число целое, номер загружаемого поля,
- **ЗначениеПоля** – Строка, загружаемое значение.

Возвращаемое значение – логическое. **Истина**, если операция прошла без ошибок.

СИСТЕМНЫЕ ФУНКЦИИ

ДОБАВИТЬСООБЩЕНИЕ(КодСообщения[, ТекстСообщения])
ADDMESSAGE(КодСообщения[, ТекстСообщения])

Добавляет указанное сообщение в список сообщений и ошибок отчета. Также это сообщение добавляется в журнал ошибок и сообщений системы (errorlog).

Параметры.

КодСообщения – Строка, код сообщения. Должен начинаться с W для сообщений и с E для ошибок.

ТекстСообщения – Строка, текст сообщения.

Возвращаемое значение – нет.

ДАТАКОНСТАНТ([Дата])
CONSTANTSDATE([Дата])

Устанавливает дату для определения значений периодических констант.

Параметры.

Дата – Дата, на которую будут вычислены значения используемых констант. Если параметр не указан, то будет использована текущая дата.

Возвращаемое значение – нет.

ВЫЧИСЛИТЬ(Программа[, ТипЗначения[,ЗначениеПоУмолчанию]])
EVALUATE(Программа[, ТипЗначения[, ЗначениеПоУмолчанию]])

Выполняет набор команд переданный в качестве аргумента. Команды передаются в виде строки текста, который анализируется и выполняется. Выполнение производится в текущем окружении, т.е. доступны все текущие переменные и контексты. Также, если в тексте Программы определяются новые переменные или изменяются значения текущих переменных, производятся какие-либо действия с контекстами, то все эти изменения отражаются на вызывающей программе. Размер программы не ограничен. Функция возвращает значение последнего вычисленного выражения. Если указан необязательный параметр ТипЗначения, то значение последнего вычисленного выражения преобразуется к указанному типу и возвращается в качестве результата вычисления.

Параметры.
Программа - Строковое выражение, содержащее текст выполняемой программы.
ТипРезультата - Символ, обозначающий тип возвращаемого результата.
Допустимые значения параметра:
'C' - символьный результат,
'I' - число целое,
'F' или 'B' - число вещественное,
'L' - логическое,
'D' - дата и время.
ЗначениеПоУмолчанию- Любое значение. Будет являться результатом, в случае, если во время вычисления возникли ошибки.
Возвращаемое значение - зависит от выполняемой программы и является значением последнего вычисленного выражения Программы, либо значением по умолчанию.

ПЕРЕМЕННАЯ(ИмяПеременной [, ЗначениеПоУмолчанию])
VARIABLE(ИмяПеременной [, ЗначениеПоУмолчанию])
Возвращает значение указанной переменной, константы или понятия. В случае, если переменная не определена, то ищет значение среди констант и понятий. Если ничего не было найдено, то возвращает значение по умолчанию. Не вызывает ошибок, если переменная, константа и понятия с указанным именем не определены. Не вычисляет значений выражений. Должно быть передано только имя переменной, константы или понятия.
Параметры.
ИмяПеременной - Строка. Имя переменной.
ЗначениеПоУмолчанию- Любое значение. Будет являться результатом, в случае, если указанной переменной нет.
Возвращаемое значение - зависит от выполняемой программы и является значением последнего вычисленного выражения Программы, либо значением по умолчанию.

СООБЩЕНИЕ([ТекстСообщения[, Заголовок[, Флаги]]])
MESSAGEBOX([ТекстСообщения[, Заголовок[, Флаги]]])
Выводит диалоговое окно с указанным ТекстомСообщения и Заголовком. Флаги могут содержать информацию о количестве и содержании доступных кнопок.
Параметры.
ТекстСообщения - Строка, содержащая текст выводимого сообщения. Строка может содержать символы перевода строки (0x0D) для отображения нескольких строк текста.
Заголовок - Строка, содержащая текст, который будет выведен в заголовок окна. Если параметр не указан, то в заголовок будет выведено слово Сообщение.
Флаги - Число, указывающее, какой набор кнопок будет присутствовать в окне. Если параметр не указан, то будет использовано значение 0. Допустимые значения параметра:
0 - одна кнопка OK,
1 - две кнопки OK и Отмена,
2 - три кнопки Прервать, Отменить и Пропустить,
3 - три кнопки Да, Нет и Отмена,
4 - две кнопки Да и Нет,
5 - две кнопки Повторить и Отменить.
Возвращаемое значение - Число, соответствующее нажатой пользователем кнопке. Возможные значения:
1 - кнопка OK,
2 - кнопка Отмена,
3 - кнопка Прервать,
4 - кнопка Повторить,
5 - кнопка Пропустить,
6 - кнопка Да,
7 - кнопка Нет.

ИНДИКАТОР([Заголовок/Индикатор[,База/Подзаголовок[,Подзаголовок/Заголовок]]])

PROGRESSBAR([Заголовок/Индикатор[,База/Подзаголовок[,Подзаголовок/Заголовок]]])
Выводит и управляет отображением индикатора хода выполнения работы. В зависимости от переданных параметров, функция выполняет следующие действия. Если первый параметр Заголовок имеет строковое значение, то функция инициализирует новый индикатор. В этом случае, второй параметр должен указывать общее количество обрабатываемых данных, принимаемое за 100%. Третий параметр не используется. Если первый параметр числовой, то функция обновляет текущий выведенный индикатор, изменяя его отображение. При этом второй параметр указывает на подзаголовок, выводимый непосредственно над полосой индикатора и дающий словесное описание хода работы. Третий параметр может изменить основной заголовок индикатора. Если параметры не указаны, то индикатор убирается и его работа считается завершенной.
Параметры.
Заголовок - Строка, выводимая в заголовок окна индикатора.
Индикатор - Число, на основании которого рассчитывается процентное выполнение работы. Процент вычисляется относительно Базы, указанной при инициализации индикатора.
База - Число, общее количество обрабатываемых данных, принимаемое за 100%.
Подзаголовок- Строка, подзаголовок, дающий словесное описание хода работы.
Возвращаемое значение - нет.

СИСТЕМНОЕСООБЩЕНИЕ([ТекстСообщения])
SYSTEMMESSAGE([ТекстСообщения])
Выводит окно с текстом системного сообщения. Окно с текстом системного сообщения - это окно, которое выводится в правом верхнем углу активного окна приложения, содержащее одну строку текста, который информирует пользователя о выполняемых в данный момент действиях. Данное окно не взаимодействует с пользователем и не требует его действий.
Параметры.
ТекстСообщения - Строка, текст сообщения, которое будет выведено. Если параметр не указан, окно с системным сообщением будет убрано с экрана.
Возвращаемое значение - нет.

УНИКАЛЬНОЕИМЯ()
UNIQUENAME()
Функция возвращает строку, которая может быть использована в качестве уникального имени. Данное имя может быть использовано, например, для создания временных таблиц.
Параметры.
нет.
Возвращаемое значение - Строка, содержащая уникальную последовательность символов.

ОТПРАВИТЬСООБЩЕНИЕ(ДескрипторОкна, Сообщение[, wParam[, lParam]])
SENDMESSAGE(ДескрипторОкна, Сообщение[, wParam[, lParam]])
Посылает указанное Сообщение указанному окну. Используется системная функция SendMessage. В отличие от ОтправитьСообщение, функция возвращается только после обработки сообщения указанным окном.
Параметры.
ДескрипторОкна - Число целое, дескриптор окна, которому посылается сообщение.
Сообщение - Число целое, сообщение для посылки.
wParam - Число целое, первый параметр сообщения.
lParam - Число целое, второй параметр сообщения.
Возвращаемое значение - Число целое, результат обработки сообщения функцией окна получателя сообщения.

ПОСЛАТЬСООБЩЕНИЕ(ДескрипторОкна, Сообщение[, wParam[, lParam]])
POSTMESSAGE(ДескрипторОкна, Сообщение[, wParam[, lParam]])

Отправляет указанное Сообщение указанному окну. Используется системная функция PostMessage. В отличие от ПослатьСообщение, функция возвращается немедленно после отправки сообщения.
Параметры.
ДескрипторОкна - Число целое, дескриптор окна, которому отправляется сообщение.
Сообщение - Число целое, сообщение для отправки.
wParam - Число целое, первый параметр сообщения.
lParam - Число целое, второй параметр сообщения.
Возвращаемое значение - Логическое. **Истина**, если сообщение успешно отправлено. Иначе - **Ложь**.

СОЗДАТЬОБЪЕКТ(ИмяКласса[, КонтекстИсполнения])
CREATEOBJECT(ИмяКласса[, КонтекстИсполнения])
Создает экземпляр COM объекта и возвращает ссылку на его IDispatch интерфейс.
Параметры.
ИмяКласса - Строка, имя класса объекта, зарегистрированное в реестре Windows.
КонтекстИсполнения- Контекст, в котором будет работать новый объект.
Допустимые значения для данного параметра:
CLSCTX_INPROC_SERVER 1,
CLSCTX_INPROC_HANDLER 2,
CLSCTX_LOCAL_SERVER 4,
CLSCTX_REMOTE_SERVER 16,
CLSCTX_NO_CODE_DOWNLOAD 400
По умолчанию используется значение CLSCTX_LOCAL_SERVER.
Возвращаемое значение - Ссылка на объект. Если создание объекта закончилось неудачей, то будет возвращена ссылка на пустой объект.

ВЫПОЛНИТЬ(ИмяФайла[, Параметры[, РабочийКаталог[, Действие[, Флаги]]]])
SHELLEXECUTE(ИмяФайла[, Параметры[, РабочийКаталог[, Действие[, Флаги]]]])
Запускает указанный файл ИмяФайла на выполнение. В качестве имени файла могут быть указаны исполняемые файлы, файлы документов и другие, зарегистрированные в системе файлы. Можно указать строку параметров для запускаемого приложения и его РабочийКаталог. Для документов можно указать действие, которое требуется выполнить. В качестве Действия могут быть указаны следующие: **open** - открыть документ, **print** - распечатать документ, **edit** - редактировать документ. И другие действия, зарегистрированные в системе. Дополнительно можно запретить отображать окно запускаемого приложения и указать на необходимость ждать окончания его работы.
Параметры.
ИмяФайла - Строка содержащая имя файла, который необходимо выполнить.
Параметры - Строка содержащая список параметров, через запятую, которые будут переданы запускаемой программе.
РабочийКаталог- Строка содержащая путь к рабочему каталогу запускаемого файла.
Команда - Строка, содержащая название действия, которое необходимо выполнить.
Допустимые значения для данного параметра:
Open Открыть, запустить на выполнение,
Print Печатать,
Edit Открыть документ в редакторе,
другие зарегистрированные команды. По умолчанию используется значение Open.
Флаги - Число целое. Допустимые значения для данного параметра:

- 0 - Показывать окно приложения и не ждать окончания его работы,
- 1 - Не показывать окно приложения и не ждать окончания его работы,
- 2 - Показывать окно приложения и ждать окончания его работы,
- 3 - Не показывать окно приложения и ждать окончания его работы.

Возвращаемое значение - Логическое, **Истина**, в случае успешного выполнения, иначе - **Ложь**.

ЖУРНАЛСООБЩЕНИЙ(ИмяФайлаЖурнала[,ЗаписыватьВБазу[,Флаги[,КоличествоСообщений]]])
MESSAGELOG(ИмяФайлаЖурнала[,ЗаписыватьВБазу[,Флаги[,КоличествоСообщений]]])
Устанавливает параметры ведения журнала ошибок и сообщений.
Параметры.
ИмяФайлаЖурнала - Строка, имя файла, в который будут записываться сообщения. Если в качестве параметра передана пустая строка, то запись сообщений в файл не будет производиться. Если указана логическая константа **ЛОЖЬ**, то текущее имя файла журнала изменено не будет.
ЗаписыватьВБазу - Логическое. **ИСТИНА**, если требуется вести запись сообщений в таблицу errorlog, или **ЛОЖЬ**, если не требуется.
Флаги - Число целое. Допустимые значения для данного параметра:
Пропускать системные предупреждения 1,
Пропускать все предупреждения 2,
Пропускать предупреждения и ошибки SQL 4,
Пропускать все ошибки 8.
КоличествоСообщений- Число целое, указывающее максимально возможное количество сообщений, которые будут записаны.
Возвращаемого значения нет.

ОЧИСТИТЬМАССИВ(ИмяМассива)
CLEARARRAY(ИмяМассива)
Очищает содержимое массива и делает количество элементов равным нулю. Если указанного массива нет, то функция ничего не делает.
Параметры.
ИмяМассива - Строка, имя массива, который необходимо очистить.
Возвращаемое значение - нет.

КОЛИЧЕСТВОЭЛЕМЕНТОВ(ИмяМассива)
ARRAYCOUNT(ИмяМассива)
Возвращает количество элементов в указанном массиве. Если массива нет, то -1.
Параметры.
ИмяМассива - Строка, имя массива.
Возвращаемое значение - число целое. Количество элементов массива или -1.

СФОРМИРОВАТЬ_PDF417(Текст,ТипРезультата,[УровеньКоррекцииОшибок[,КоличествоКолонок]])
PDF417(Текст,ТипРезультата,[УровеньКоррекцииОшибок[,КоличествоКолонок]])
Функция формирует текст или двоичные данные для отображения 2D штрихового кода в формате PDF417. Функция может возвращать данные в двух видах - в виде двоичного массива, состоящего из кодовых слов в формате PDF417 и в виде строки текста, которая может быть отображена в виде штрихового кода при использовании специального шрифта. Так же могут быть указаны требуемый уровень коррекции ошибок и количество кодовых слов в одной строке. Если уровень коррекции ошибок и/или количество кодовых слов в одной строке не указаны, то данные параметры подбираются автоматически на основании стандартных рекомендаций.
Параметры.
Текст - Строка, которую необходимо закодировать.
Формат - Символ 'B', если необходимо сформировать двоичный массив или 'S', если необходима строка для отображения специальным шрифтом.
УровеньКоррекцииОшибок - Число от 1 до 8. Чем выше уровень, тем больше кодовых слов отводится для коррекции ошибок.
КоличествоСтолбцов - Число от 1 до 30 равное желаемому количеству кодовых слов в одной строке штрихового кода.
Возвращаемое значение - Двоичный массив или строка. В случае возникновения ошибки - пустый. Код ошибки возвращается в переменной _ERRORCODE. Возможные значения ошибок: -1: Ошибка распределения памяти, 1: исходная строка пуста, 2: исходная строка содержит слишком много данных (более 928 кодовых слов), 3: Количество кодовых слов на строку слишком мало, 10: уровень коррекции ошибок был понижен, чтобы уместиться в 928 кодовых слов (предупреждение).

ФОРМАТ(СтрокаФормата[, Параметр1[, ...]])
FORMAT(СтрокаФормата[, Параметр1[, ...]])
Возвращает строку, полученную из значений разных типов, отформатированных согласно заданному шаблону. Этот шаблон определяется составленной по специальным правилам строкой (форматной строкой).
Параметры.
СтрокаФормата – Строка, определяющая шаблон для преобразования остальных параметров.
Параметр1,... – Параметры одного из следующих типов:

- число целое,
- число вещественное,
- строка.

Возвращаемое значение – Строка, полученная путем форматирования переданных параметров по указанному шаблону.

ЯВЛЯЕТСЯСИМВОЛОМ(Строка[, НомерСимвола])
ISSYMBOL(Строка[, НомерСимвола])
Проверяет, является ли указанный символ строки символом, допустимым в идентификаторах, кроме цифр. Допустимыми символами являются буквы, а так же символы '_', '#' и '@'.
Параметры.
Строка – Строка, в которой содержится проверяемый символ.
НомерСимвола – Число целое, номер проверяемого символа в строке.
Возвращаемое значение – Логическое, если указанный символ является допустимым символом, то **ИСТИНА**, иначе **ЛОЖЬ**.

ЯВЛЯЕТСЯЦИФРОЙ(Строка[, НомерСимвола])
ISDIGIT(Строка[, НомерСимвола])
Проверяет, является ли указанный символ строки цифрой.
Параметры.
Строка – Строка, в которой содержится проверяемый символ.
НомерСимвола – Число целое, номер проверяемого символа в строке.
Возвращаемое значение – Логическое, если указанный символ является цифрой, то **ИСТИНА**, иначе **ЛОЖЬ**.

ЯВЛЯЕТСЯИДЕНТИФИКАТОРОМ(Строка)
ISIDENTIFIER(Строка)
Проверяет, может ли быть указанная строка идентификатором. Идентификатором может быть строка, которая содержит только допустимые символы или цифры и начинается с допустимого символа, не цифры.
Параметры.
Строка – Строка, для которой выполняется проверка.
Возвращаемое значение – Логическое, если указанная строка может быть идентификатором, то **ИСТИНА**, иначе **ЛОЖЬ**.

ПОТОК(Выражение, Ожидать[, ВремяОжидания[, Сообщение[, ИмяПеременной1[,...]]])
THREAD(Выражение, Ожидать[, ВремяОжидания[, Сообщение[, ИмяПеременной1[,...]]])
Вычисляет значение выражения в отдельном потоке. Первым параметром передается текст выражения. Второй параметр указывает, на необходимость дожидаться окончания выполнения потока. Если выполняется ожидание окончания выполнения потока, то в процессе ожидания обрабатывается очередь сообщений Windows. Остальные параметры должны быть именами переменных, которые необходимо перенести в новый поток. Для вычисления выражения создается новый Вычислитель, который запускается в отдельном потоке выполнения команд. Если не производится ожидания окончания работы потока, то создается новое соединение с источником данных, которое наследует параметры текущего соединения.
Параметры.
Выражение – Строка, вычисляемое выражение / набор команд.

Ожидать – Логическое, если Истина, то будет организован цикл ожидания окончания выполнения потока.
ВремяОжидания – Число целое. Если больше нуля, то ожидание продлится в пределах указанного количества миллисекунд. По истечении интервала выполнение программы продолжится, а запущенному потоку будет подан сигнал завершения. Используется только, если требуется ожидать окончания выполнения потока.
Сообщение – Строка. Если указана не пустая строка, то перед выполнением потока будет выведено окно с сообщением и кнопкой «Отмена». При нажатии на кнопку выполнение потока будет прервано, даже если время ожидания не истекло. Сообщение может содержать в себе три части, разделенные символом | (вертикальная черта). Первая часть строки будет текстом сообщения, вторая – заголовком окна, а третья текстом кнопки. Если второй разделитель будет присутствовать, а текст кнопки не будет указан, то окно будет без кнопки. Используется только, если требуется ожидать окончания выполнения потока.
Параметр1 – Строка, имя переменной, которую необходимо определить в новом вычислителе перед началом выполнения.
Возвращаемое значение – если осуществляется ожидание окончания выполнения потока, то возвращается результат вычисления выражения. Иначе – логическое. **ИСТИНА**, если поток был успешно запущен. В случае ошибки возвращается **ЛОЖЬ**.

ОБРАБОТАТЬСОБЫТИЯ( )
PROCESSEVENTS( )
Функция обрабатывает текущую очередь сообщений Windows, пропуская сообщения от мыши и клавиатуры.
Параметры.
нет.
Возвращаемое значение – Логическое **ИСТИНА**.

СИСТЕМНОЕВРЕМЯ( )
TICKCOUNT( )
Возвращает количество миллисекунд после запуска операционной системы. Выполняется системный вызов GetTickCount( ).
Параметры.
нет.
Возвращаемое значение – Число целое.

ФОРМА(ИДФормы, СписокФилиалов[, Параметр1, [Параметр2 [...]]])
CREATEFORM(ИДФормы, СписокФилиалов[, Параметр1, [Параметр2 [...]]])
Создает экземпляр COM объекта формы и возвращает ссылку на его IDispatch интерфейс.
Параметры.
ИДФормы – Строка, идентификатор шаблона формы, зарегистрированный в реестре шаблонов.
СписокФилиалов – Строка, список кодов филиалов через запятую или пробел, с которыми должна работать создаваемая форма.
Параметр1...n – Параметры, передаваемые методу инициализации формы.
Возвращаемое значение – Ссылка на объект формы. Если создание объекта закончилось неудачей, то будет возвращена ссылка на пустой объект.

ШИФРОВАНИЕ(Сертификат, Данные, Операция[, ОткрепленнаяПодпись[, ОригиналДанных[, Хранилище]]]])
CRYPTOGRAPHY(Сертификат, Данные, Операция[, ОткрепленнаяПодпись[, ОригиналДанных[, Хранилище]]]])
Выполняет указанную криптографическую операцию с данными, используя сертификат. Возвращает массив двоичных данных. Сертификат передается в виде хэша MD5 от Издатель + СерийныйНомер в кодировке base 64. Если сертификат не передан, то будет предложено выбрать из установленных в системе сертификатов.
Параметры.

Сертификат - Строка, идентификатор сертификата. Идентификатор - это хэш MD5 от Издателя сертификата, дополненного серийным номером. Может быть получен функцией **СПИСОКСЕРТИФИКАТОВ**( ).
Данные - Строка или двоичный массив, с которым необходимо выполнить операцию.
Операция - Число целое. Допустимые значения:
1 - Подписать данные ЭЦП;
2 - Зашифровать данные;
3 - Подписать и зашифровать данные;
4 - Проверить ЭЦП;
5 - Расшифровать данные;
6 - Расшифровать данные и проверить ЭЦП.
7 - Получить дамп **сертификата**. При этом, параметр **Данные** не используется.
Значение по умолчанию 1.
ОткрепленнаяПодпись- Логическое. Если **Истина**, то будет сформирована открепленная ЭЦП. Значение по умолчанию **Ложь**.
ОригиналДанных - Строка или массив. При проверке открепленной ЭЦП, расшифрованные данные для сравнения подписи.
Хранилище - Строка, хранилище, в котором нужно искать сертификат.
Допустимые значения:
MY - локальное хранилище пользователя;
ROOT - Корневые сертификаты;
CA - Центры сертификации;
SPC - Издатели ПО.
Значение по умолчанию **MY**.
Возвращаемое значение - для операции 4 логическое значение результат проверки ЭЦП. Для остальных - двоичный массив результат выполненной операции. В случае ошибки будет возвращен пустой массив, а переменная _ERRORCODE получит значение - 24.

СПИСОКСЕРТИФИКАТОВ(ИмяКонтекста[, Хранилище])
CERTIFICATES(ИмяКонтекста[, Хранилище])
Заполняет указанный контекст списком сертификатов из указанного хранилища. Если контекст уже существует, то он будет удален и создан новый. Структура контекста: **certid** - ИД Сертификата, **issuer** - Издатель, **subject** - Получатель, **description** - Описание, **name** - Понятное имя. Все поля текстовые. ИД сертификата получается в виде хэша MD5 от Издатель + СерийныйНомер в кодировке base 64.
Параметры.
ИмяКонтекст - Строка, имя контекста, в котором будет получен список сертификатов.
Хранилище - Строка, хранилище, из которого нужно брать сертификаты.
Допустимые значения:
MY - локальное хранилище пользователя;
ROOT - Корневые сертификаты;
CA - Центры сертификации;
SPC - Издатели ПО.
Значение по умолчанию **MY**.
Возвращаемое значение - логическое. **Истина**, если ошибок не было, иначе **Ложь**.

МД5(Строка[, Тип])
MD5(Строка[, Тип])
Функция возвращает массив байт содержащий MD5 переданной строки.
Параметры.
Строка- Строка, которая будет преобразована в MD5.
Тип- Строка, допустимые значения:
'B' - функция вернет байтовый массив,
'S' - функция вернет строку.
Возвращаемое значение - массив байт или строка.

ХЭШ256(Данные[, Тип])
SHA256(Данные[, Тип])
Функция вычисляет значение хэш функции для переданных данных по алгоритму **SHA256**. Возвращает результат в виде двоичного массива или строки в кодировке **BASE64**.
Параметры.
Данные- Значение любого типа, для которого будет вычислено значение хэш функции.
Тип- Строка, допустимые значения:
'B' - функция вернет байтовый массив,
'S' - функция вернет строку в кодировке BASE64.
Возвращаемое значение - Двоичный массив или строка.

УДАЛИТЬСУЩНОСТИ(СписокИмен[,Сущности])
RELEASE(СписокИмен[,Сущности])
Удаляет из памяти перечисленные переменные, массивы и контексты.
Параметры.
СписокИмен- Строка, список имен через запятую.
Сущности- Строка. Какие сущности удалять. Допустимые типы
1. М (кириллическое) или A (латинское) – массивы
2. П или V – переменные
3. К (кириллическое) или C (латинское) – контексты

Если значение параметра не указано, то все.
Возвращаемое значение - нет.
Пример использования.
```
УДАЛИТЬСУЩНОСТИ( "Переменная, КОНТЕКСТ", "МК" ); // Если Переменная не массив, то удалена не будет
УДАЛИТЬСУЩНОСТИ( "Переменная, КОНТЕКСТ" );
```

УУИД( )
UUID( )
Функция генерирует и возвращает в виде строки новое значение UUID (уникального универсального идентификатора).
Возвращаемое значение – Строка.

ЭЛЕМЕНТОВВМАССИВЕ(ИмяМассива)
ACOUNT(ИмяМассива)
Функция возвращает количество элементов в массиве.
Параметры.
ИмяМассива - Строка, имя массива, для которого необходимо определить количество элементов.
Возвращаемое значение - число целое, равное количеству элементов в массиве, либо 0, если указанного массива нет.

ФУНКЦИИ ДЛЯ РАБОТЫ С СОКЕТАМИ

СОКЕТСОЕДИНИТЬ(ИмяСервера | IP-адрес[, Порт[, ВестиЖурнал]])
SOCKETCONNECT(ИмяСервера | IP-адрес[, Порт[, ВестиЖурнал]])
Функция выполняет соединение с указанным сервером через указанный порт. Возвращает идентификатор соединения.
Параметры.
ИмяСервера - Строка, имя сервера, для преобразования в IP адрес посредством DNS или файла hosts.
IP-адрес - Строка, IP адрес сервера, с которым необходимо произвести соединение.
Порт - Число, номер порта, на который необходимо произвести соединение. Если параметр не указан, то номер порта должен быть указан в первом параметре в формате **ИмяСервера:Порт.**
ВестиЖурнал - Логическое, если Истина, то все принятые и переданные данные будут записываться в журнал.

Возвращаемое значение - Число целое. Идентификатор соединения в случае успешного соединения. Иначе - 0.

СОКЕТПЕРЕДАТЬ(Соединение, Данные[, КоличествоБайт])
SOCKETSEND(Соединение, Данные[, КоличествоБайт])
Функция выполняет отправку данных через указанное соединение.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **СокетСоединить**().
Данные - Параметр любого типа, значение которого необходимо передать.
КоличествоБайт - Число целое, количество байт, которое необходимо передать. Если параметр не указан, то будут переданы все данные значения второго параметра.
Возвращаемое значение - Логическое. ИСТИНА, если отправка завершилась успешно. Иначе - ЛОЖЬ.

СОКЕТПОЛУЧИТЬ(Соединение,КоличествоБайт|СимволыОграничители[,ТипДанных[,ВремяОжидания]])
SOCKETRECIEVE(Соединение,КоличествоБайт|СимволыОграничители[,ТипДанных[,ВремяОжидания]])
Функция выполняет получение из указанного соединения, указанного количества байт или всех байт, пока не встретит указанной строки символов ограничителей.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **СокетСоединить**( ).
КоличествоБайт - Число целое, количество байт, которое необходимо принять.
СимволыОграничители - Строка, получение которой из входного потока заканчивает прием данных.
ТипДанных - Строка состоящая из одного символа, определяющего тип возвращаемых функцией данных. Допустимые значения для данного параметра:
S строка,
I число целое,
F число вещественное,
D дата и время,
L логическое,
B двоичные данные.
По умолчанию используется значение B.
ВремяОжидания - Число целое, количество миллисекунд, по истечении которых прием данных будет прерван. Если параметр не указан, то будет использовано значение 10000.
Возвращаемое значение - Зависит от указанного типа значения. Содержит принятые данные преобразованные в указанный тип данных.

СОКЕТЗАКРЫТЬ(Соединение)
SOCKETCLOSE(Соединение)
Функция закрывает указанное соединение и освобождает все связанные с ним ресурсы.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **СокетСоединить**().
Возвращаемое значение - Логическое. ИСТИНА, если операция завершена успешно. Иначе - ЛОЖЬ.

ФУНКЦИИ ДЛЯ РАБОТЫ С FTP СЕРВЕРАМИ

ФТПСОЕДИНИТЬ(ИмяСервера|IP,ИмяПользователя,Пароль[,ПассивныйРежим[,Порт])
FTPCONNECT(ИмяСервера|IP-адрес,ИмяПользователя,Пароль[,ПассивныйРежим[,Порт])

Функция выполняет соединение с указанным FTP сервером, используя для этого указанное имя пользователя и пароль. Возвращает идентификатор соединения.
Параметры.
ИмяСервера - Строка, имя сервера, для преобразования в IP адрес посредством DNS или файла hosts.
IP-адрес - Строка, IP адрес сервера, с которым необходимо произвести соединение.
ИмяПользователя - Строка, имя пользователя для авторизации на сервере.
Пароль - Строка, пароль пользователя для авторизации на сервере.
ПассивныйРежим - Логическое. Если Истина, то будет использован пассивный режим работы сервера. Если параметр не указан, то будет использовано значение Истина.
Порт - Число, номер порта, на который необходимо произвести соединение. Если параметр не указан, то номер порта должен быть указан в первом параметре в формате ИмяСервера:Порт. Если в имени сервера порт так же не указан, то будет использован 21-й порт.
Возвращаемое значение - Число целое. Идентификатор соединения в случае успешного соединения. Иначе - 0.

ФТПЗАКРЫТЬ(Соединение)
FTPCLOSE(Соединение)
Функция закрывает указанное соединение и освобождает все связанные с ним ресурсы.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
Возвращаемое значение - Логическое. ИСТИНА, если операция завершена успешно. Иначе - ЛОЖЬ.

ФТППЕРЕДАТЬФАЙЛ(Соединение, ИмяЛокальногоФайла, ИмяУдаленногоФайла)
FTPPUTFILE(Соединение, ИмяЛокальногоФайла, ИмяУдаленногоФайла)
Функция передает указанный локальный файл с именем ИмяЛокальногоФайла на FTP сервер и записывает его в текущий каталог FTP сервера под именем ИмяУдаленногоФайла.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
ИмяЛокальногоФайла - Строка, путь и имя файла на локальном компьютере или в локальной сети, который необходимо передать на FTP сервер.
ИмяУдаленногоФайла - Строка, имя без пути, под которым необходимо сохранить переданный файл на FTP сервере.
Возвращаемое значение - Логическое. ИСТИНА, если операция завершена успешно. Иначе - ЛОЖЬ.

ФТППОЛУЧИТЬФАЙЛ(Соединение, ИмяУдаленногоФайла, ИмяЛокальногоФайла)
FTPGETFILE(Соединение, ИмяУдаленногоФайла, ИмяЛокальногоФайла)
Функция передает указанный файл из текущего каталога на FTP сервере с именем ИмяУдаленногоФайла на локальный компьютер пользователя и сохраняет его с указанным именем по указанному пути.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
ИмяУдаленногоФайла - Строка, имя файла без пути в текущей папке на FTP сервере, который необходимо передать на локальный компьютер.
ИмяЛокальногоФайла - Строка, путь и имя на локальном компьютере или в локальной сети, под которым необходимо сохранить полученный файл.
Возвращаемое значение - Логическое. ИСТИНА, если операция завершена успешно. Иначе - ЛОЖЬ.

ФТППОЛУЧИТЬФАЙЛПОШАБЛОНУ(Соединение, ШаблонИмениФайла, ИмяЛокальнойПапки)

FTPGETFILEBYTEMPLATE(Соединение, ШаблонИмениФайла, ИмяЛокальнойПапки)
Функция передает файлы из текущего каталога на FTP сервере с именами удовлетворяющими указанному шаблону на локальный компьютер пользователя и сохраняет их в указанной папке.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
ШаблонИмениФайла - Строка, шаблон имён файлов без пути в текущей папке на FTP сервере, которые необходимо передать на локальный компьютер.
ИмяЛокальнойПапки - Строка, путь и имя папки на локальном компьютере или в локальной сети, в которой необходимо сохранить полученные файлы.
Возвращаемое значение - Логическое. ИСТИНА, если операция завершена успешно. Иначе - ЛОЖЬ.

ФТПУДАЛИТЬФАЙЛ(Соединение, ШаблонИмениФайла)
FTPDELETEFILE(Соединение, ШаблонИмениФайла)
Функция удаляет файлы из текущей папки на FTP сервере, имена которых удовлетворяют указанному шаблону.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
ШаблонИмениФайла - Строка, шаблон имён файлов без пути в текущей папке на FTP сервере, которые необходимо удалить.
Возвращаемое значение - Логическое. ИСТИНА, если операция завершена успешно. Иначе - ЛОЖЬ.

ФТПКОЛИЧЕСТВОФАЙЛОВ(Соединение[, ШаблонИменФайлов])
FTPFILESCOUNT(Соединение[, ШаблонИменФайлов])
Функция подсчитывает количество файлов в текущей папке на FTP сервере, имена которых удовлетворяют указанному шаблону. Если шаблон имён файлов не указан, то функция возвращает общее количество файлов в текущей папке на FTP сервере.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
ШаблонИмениФайла - Строка, шаблон имён файлов без пути в текущей папке на FTP сервере, которые необходимо посчитать.
Возвращаемое значение - Число целое, равное количеству найденных файлов, в случае успешного завершения и -1 в случае возникновения ошибки.

ФТПУСТАНОВИТЬКАТАЛОГ(Соединение, ИмяКаталога)
FTPSETCURRENTDIR(Соединение, ИмяКаталога)
Функция устанавливает указанный каталог в качестве текущего на FTP сервере.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
ИмяКаталога - Строка, имя каталога, который необходимо сделать текущим на FTP сервере.
Возвращаемое значение - Логическое. ИСТИНА, если операция завершена успешно. Иначе - ЛОЖЬ.

ФТПТЕКУЩИЙКАТАЛОГ(Соединение)
FTPGETCURRENTDIR(Соединение)
Функция возвращает имя текущего каталога на FTP сервере.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
Возвращаемое значение - Строка, содержащая имя текущего каталога на FTP сервере. В случае возникновения ошибки возвращается пустая строка.

ФТПОШИБКА(Соединение)
FTPERROR(Соединение)
Функция возвращает истину, если при выполнении последней операции с FTP сервером возникла ошибка.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
Возвращаемое значение - Логическое, ИСТИНА, если при выполнении последней операции с FTP сервером возникла ошибка. Иначе - ЛОЖЬ.

ФТПТЕКСТОШИБКИ(Соединение)
FTPERRORTEXT(Соединение)
Функция возвращает строку, содержащую описание ошибки, возникшей при выполнении последней операции с FTP сервером.
Параметры.
Соединение - Число, идентификатор соединения, возвращенный функцией **ФТПСоединить()**.
Возвращаемое значение - Строка, описывающая возникшую ошибку при выполнении последней операции с FTP сервером. В случае отсутствия ошибок, будет возвращена строка **'Нет ошибки'**

ФТПСОЗДАТЬКАТАЛОГ(Соединение, ИмяКаталога)
FTPCREATEDIRECTORY(Соединение, ИмяКаталога)
Функция создает новый каталог с указанным именем в текущем каталоге на FTP сервере.
Параметры.
Соединение- Число, идентификатор соединения, возвращенный функцией **ФТПСоединить**().
ИмяКаталога- Строка, имя каталога, который необходимо создать.
Возвращаемое значение - Логическое. **Истина**, если операция завершена успешно. Иначе - **Ложь**.

ФТПУДАЛИТЬКАТАЛОГ(Соединение, ИмяКаталога)
FTPDELETEDIRECTORY(Соединение, ИмяКаталога)
Функция удаляет каталог с указанным именем из текущего каталога на FTP сервере.
Параметры.
Соединение- Число, идентификатор соединения, возвращенный функцией **ФТПСоединить**().
ИмяКаталога- Строка, имя каталога, который необходимо удалить.
Возвращаемое значение - Логическое. **Истина**, если операция завершена успешно. Иначе - **Ложь**.

ФУНКЦИИ ДЛЯ РАБОТЫ С HTTP СЕРВЕРАМИ

HTTPСОЕДИНИТЬ(ИмяСервера | IP-адрес[, Порт[, Защищенное[, ФайлЖурнала]]])
HTTPCONNECT(ИмяСервера | IP-адрес[, Порт[, Защищенное[, ФайлЖурнала]]])
Функция выполняет соединение с указанным **HTTP** сервером. Выбор того, будет ли установлено защищенное соединение, производится на основании префикса **http** | **https**, либо на основании значения параметра Защищенное, если он указан. Возвращает идентификатор соединения.
Параметры.
ИмяСервера- Строка, имя сервера, для преобразования в IP адрес посредством DNS или файла hosts.
IP-адрес- Строка, IP адрес сервера, с которым необходимо произвести соединение.
Порт- Число, номер порта, на который необходимо произвести соединение. Если параметр не указан, то номер порта может быть указан в первом параметре в формате ИмяСервера:Порт. Если в имени сервера порт так же не указан, то будет использован 80-й порт.
Защищенное- Логическое. Указывает, надо ли пытаться установить защищенное соединение.

ФайлЖурнала- Строка. Имя файла, для записи диалога с HTTP сервером.
Возвращаемое значение - Число целое. Идентификатор соединения в случае успешного соединения. Иначе - 0.
Пример.

```
Соединение     = HTTPСОЕДИНИТЬ( "https://example.com" );
```

HTTPЗАКРЫТЬ(Соединение)
HTTPCLOSE(Соединение)
Функция закрывает указанное **HTTP** соединение и освобождает занятые ресурсы.
Параметры.
Соединение- Число, номер соединения, возвращенный функцией **HTTPСОЕДИНИТЬ**( ).
Возвращаемое значение - нет,

HTTPПОЛУЧИТЬ(Соединение, Ресурс[, Заголовки[, ТолькоЗаголовки[, ТипРезультата]])
HTTPGET(Соединение, Ресурс[, Заголовки[, ТолькоЗаголовки[, ТипРезультата]])
Функция выполняет **HTTP** запрос **GET** и возвращает ответ сервера.
Параметры.
Соединение- Число, номер соединения, возвращенный функцией **HTTPСОЕДИНИТЬ**( ).
Ресурс- Строка, запрашиваемый ресурс. Ресурс может содержать указание на **HTTP** сервер. В таком случае будет произведено повторное подключение к новому серверу. В противном случае, запрос будет отправлен серверу, указанному при подключении.
Заголовки- Строка, имя массива строк, содержащего дополнительные заголовки, которые будут переданы серверу. Если для заголовка не будет указано значение, то такой заголовок будет исключен из передачи серверу. После имени массива через двоеточие может быть указано количество элементов, которое нужно передать. Например, "***Заголовки:5***" будет означать, что необходимо передать не все строки из массива, а только первые пять.
ТолькоЗаголовки- Логическое. Если указано значение Истина, то будут возвращены только поля заголовков.
ТипРезультата- Строка, определяющая тип возвращаемого результата.
Допустимые значения:
- **'S'** - строковое значение;
- **'B'** - двоичный массив.

По умолчанию, используется значение **S**.
Возвращаемое значение - Строка или двоичный массив. Ответ сервера на запрос. Если сервер вернул ошибку, то в переменной **_RESPONSECODE** будет код ошибки.

HTTPПОСЛАТЬ(Соединение, Ресурс, Текст | ДвоичныеДанные[, Заголовки[, ТипРезультата]])
HTTPPOST(Соединение, Ресурс, Текст | ДвоичныеДанные[, Заголовки[, ТипРезультата]])
Функция выполняет **HTTP** запрос **POST** и возвращает ответ сервера.
Параметры.
Соединение- Число, номер соединения, возвращенный функцией **HTTPСОЕДИНИТЬ**( ).
Ресурс- Строка, запрашиваемый ресурс.
Текст- Строка, которая будет передана серверу в качестве тела запроса.
ДвоичныеДанные- Двоичные данные, которые будут переданы серверу в качестве тела запроса.
Заголовки- Строка, имя массива строк, содержащего дополнительные заголовки, которые будут переданы серверу. Если для заголовка не будет указано значение, то такой заголовок будет исключен из передачи серверу. После имени массива через двоеточие может быть указано количество элементов, которое нужно передать. Например, "***Заголовки:5***" будет означать, что необходимо передать не все строки из массива, а только первые пять.
ТипРезультата- Строка, определяющая тип возвращаемого результата.
Допустимые значения:
- **'S'** - строковое значение;
- **'B'** - двоичный массив.

По умолчанию, используется значение **S**.

Возвращаемое значение - Строка или двоичный массив. Ответ сервера на запрос. Если сервер вернул ошибку, то в переменной **_RESPONSECODE** будет код ошибки.

HTTPРАЗМЕСТИТЬ(Соединение, Ресурс, Текст | ДвоичныеДанные[, Заголовки])
HTTPPUT(Соединение, Ресурс, Текст | ДвоичныеДанные[, Заголовки])
Функция выполняет **HTTP** запрос **PUT** и возвращает ответ сервера.
Параметры.
Соединение- Число, номер соединения, возвращенный функцией **HTTPСОЕДИНИТЬ**( ).
Ресурс- Строка, запрашиваемый ресурс.
Текст- Строка, которая будет передана серверу в качестве тела запроса.
ДвоичныеДанные- Двоичные данные, которые будут переданы серверу в качестве тела запроса.
Заголовки- Строка, имя массива строк, содержащего дополнительные заголовки, которые будут переданы серверу. Если для заголовка не будет указано значение, то такой заголовок будет исключен из передачи серверу. После имени массива через двоеточие может быть указано количество элементов, которое нужно передать. Например, "***Заголовки:5***" будет означать, что необходимо передать не все строки из массива, а только первые пять.
ТипРезультата- Строка, определяющая тип возвращаемого результата.
Допустимые значения:
- **'S'** – строковое значение;
- **'B'** – двоичный массив.

По умолчанию, используется значение **S**.
Возвращаемое значение - Строка или двоичный массив. Ответ сервера на запрос. Если сервер вернул ошибку, то код ошибки.

HTTPУДАЛИТЬ(Соединение, Ресурс[, Заголовки[, ТипРезультата]])
HTTPDELETE(Соединение, Ресурс[, Заголовки[, ТипРезультата]])
Функция выполняет **HTTP** запрос **DELETE** и возвращает ответ сервера.
Параметры.
Соединение- Число, номер соединения, возвращенный функцией **HTTPСОЕДИНИТЬ**( ).
Ресурс- Строка, удаляемый ресурс.
Заголовки- Строка, имя массива строк, содержащего дополнительные заголовки, которые будут переданы серверу. Если для заголовка не будет указано значение, то такой заголовок будет исключен из передачи серверу. После имени массива через двоеточие может быть указано количество элементов, которое нужно передать. Например, "***Заголовки:5***" будет означать, что необходимо передать не все строки из массива, а только первые пять.
ТипРезультата- Строка, определяющая тип возвращаемого результата.
Допустимые значения:
- **'S'** – строковое значение;
- **'B'** – двоичный массив.

По умолчанию, используется значение **S**.
Возвращаемое значение - Строка или двоичный массив. Ответ сервера на запрос. Если сервер вернул ошибку, то в переменной **_RESPONSECODE** будет код ошибки.

HTTPПРОЧЕЕ(Соединение, Команда, Ресурс, Текст | ДвоичныеДанные[, Заголовки[, ТипРезультата]])
HTTPOTHER(Соединение, Команда, Ресурс, Текст | ДвоичныеДанные[, Заголовки[, ТипРезультата]])
Функция выполняет **HTTP** запрос, который указан в команде и возвращает ответ сервера.
Параметры.
Соединение- Число, номер соединения, возвращенный функцией **HTTPСОЕДИНИТЬ**( ).
Команда- Строка, выполняемая команда.
Ресурс- Строка, удаляемый ресурс.
Заголовки- Строка, имя массива строк, содержащего дополнительные заголовки, которые будут переданы серверу. Если для заголовка не будет указано значение, то такой заголовок будет исключен из передачи серверу.

ТипРезультата- Строка, определяющая тип возвращаемого результата.
Допустимые значения:

- **'S'** – строковое значение;
- **'B'** – двоичный массив.

По умолчанию, используется значение **S**.
Возвращаемое значение - Строка или двоичный массив. Ответ сервера на запрос. Если сервер вернул ошибку, то в переменной **_RESPONSECODE** будет код ошибки.

HTTPВРЕМЯОЖИДАНИЯ(Соединение, ВремяОжидания[, ТипИнтервала])
HTTPTIMEOUT(Соединение, ВремяОжидания[, ТипИнтервала])
Функция задает максимальное время ожидания ответа от **HTTP** сервера.
Параметры.
Соединение - Число целое, номер соединения, возвращенный функцией **HTTPСОЕДИНИТЬ**( ).

ВремяОдидания - Число целое. Время ожидания в указанных в типе интервала единицах.

ТипИнтервала - Число целое. Набор флагов:

- **0** – время указано в секундах;
- **1** – время указано в миллисекундах.
- **2** – интервал ожидания установки соединения, иначе общее время соединения не более чем указанный интервал.

Возвращаемое значение - Логическое. **Истина**, если номер соединения был указан верно. Иначе **Ложь**.

HTTPЗАГОЛОВКИ(Соединение, ВозвращатьЗаголовки)
HTTPHEADERS(Соединение, ВозвращатьЗаголовки)
Функция устанавливает значение параметра ВозвращатьЗаголовки для указанного соединения. Если значение параметра Истина, то после выполнения запросов **GET, POST, PUT, DELETE** заголовки, которые возвращает сервер, будут включены в результат выполнения. Иначе, все заголовки будут исключены из результата.
Параметры.
Соединение- Число, номер соединения, возвращенный функцией **HTTPСОЕДИНИТЬ**( ).
ВозвращатьЗаголовки- Логическое, указывающее, нужно ли возвращать заголовки ответов сервера.
Возвращаемое значение- Логическое. Предыдущее значение параметра.

Пример.
В примере создается контекст, содержащий номер, дату и другие реквизиты документа. Выполняется запрос к HTTP серверу, который возвращает результат в виде JSON строки. Полученный результат загружается в контекст.

```
ДОБАВИТЬКОНТЕКСТ( "SET FMTONLY ON; SELECT SPACE( 20 ) AS ndok, SPACE( 3 ) AS sklad, CONVERT( datetime, '' ) AS date, SPACE( 10 ) AS nn, CONVERT( float, 0 ) AS kolp, SPACE( 250 ) AS nnname, CONVERT( float, 0 ) AS cena", "Документы" );

Соединение      = HTTPСОЕДИНИТЬ( "https://example.com" );
Заголовки[0]    = "X-Authorization:62012fe9-a3b8-4ca2-955b-ea4a13c03888";
Заголовки[1]    = "Expect:";   // Загловок Expect: не будет передан серверу
Данные          = HTTPПОЛУЧИТЬ( Соединение, "/api/data?ltCloseDate=2017-05-29T23:59:59.000+0300&gtCloseDate=2017-01-27T00:00:00.000+0300&types=SELL", "Заголовки" ) );
// Функция загрузить загрузит полученные данные в контекст "Документы" из строки в формате JSON. При этом, если сервер вернет результат, без поля kolp, то оно будет проинициализировано значением 1. Так же, во время загрузки из номера документа будет убран префикс "A-". Данные будут загружены только из подмножества spec001. Строки из bc001 и bcd001 будут пропущены.
ЗАГРУЗИТЬ( "Документы", "JSON", Данные, "kolp= 1", "ЕСЛИ( ИмяПоля == 'ndok', ЗАМЕНИТЬ( ЗначениеПоля, 'A-', '' ),  ЗначениеПоля )", "spec001", "bc001 bcd001" );
```

ФУНКЦИИ ДЛЯ РАБОТЫ С COM ПОРТОМ

COMОТКРЫТЬ(СтрокаСоединения[, ФайлЖурнала])

COMOPEN(СтрокаСоединения[, ФайлЖурнала])
Функция открывает коммуникационный порт с параметрами указанными в строке подключения. Возвращает идентификатор порта.
Параметры.
СтрокаПодключения- строка, содержащая имя порта и параметры взаимодействия Формат строки подключения к порту:

- Наименование,Скорость,ЧислоБит,Четность,СтопБиты,КонтрольПотока
- Наименование- строка **COM1, COM2,...**
- Скорость- число **1200, 2400,...56700** или **115200**
- ЧислоБит- число **5,6,7** или 8
- Четность- латинская буква **N** (No), **E** (Even), **O** (Odd), **S** (Space), **M** (Mark)
- СтопБиты- латинская буква **O** (1 бит), **H** (1.5 бита), **T** (2 бита)
- КонтрольПотока- латинская буква **N** (NoFlowControl), **H** (Hardware Control), **S** (Software Control)

Пример
COM1,115200,8,N,O,N
ФайлЖурнала- строка, имя файла журнала, в который будет записан обмен с портом.
Возвращаемое значение - число целое. Идентификатор порта в случае успешного соединения. Иначе -1.

COMЗАКРЫТЬ(Порт)
COMCLOSE(Порт)
Функция закрывает указанный порт и освобождает занятые ресурсы.
Параметры.
Порт- число, идентификатор порта, возвращенный функцией COMОТКРЫТЬ( ).
Возвращаемое значение - нет

COMПРОЧИТАТЬ(Порт, ЧислоБайт[, Интервал])
COMREAD(Порт, ЧислоБайт[, Интервал])
Функция выполняет чтение данных из открытого COM порта. Если указан интервал ожидания, то перед чтением данных будет запущен цикл ожидания указанной длительности.
Параметры.
Порт- число, идентификатор соединения, возвращенный функцией COMОТКРЫТЬ( ).
ЧислоБайт- число, максимальное число байт, которые нужно прочитать.
Если указано -1, то будут прочитаны все доступные для чтения данные.
ВремяОжидания- число миллисекунд, которые необходимо ожидать перед попыткой чтения данных.
Возвращаемое значение - Массив байт.

COMЗАПИСАТЬ(Порт, Значение[, Интервал])
COMWRITE(Порт, Значение[, Интервал])
Функция выполняет запись указанного значения в порт. После выполнения записи производится ожидание указанное количество миллисекунд. Ожидание необходимо для более точного определения размера переданных данных. Если время ожидания не указано, то функция всегда будет возвращать 0. Но это не означает, что данные не переданы, т.к. функция не будет ожидать результатов передачи данных.
Параметры.
Порт - число, идентификатор соединения, возвращенный функцией COMОТКРЫТЬ( ).
Значение- любое значение. Перед передачей будет преобразовано в массив байт.
Интервал - число миллисекунд, которые будет ожидаться результат передачи данных.
Возвращаемое значение - число переданных байт.

Пример.

В примере открывается порт COM7. Затем ожидается 10 секунд, за которые в порту должны появиться данные. Данные читаются и выводятся пользователю. Затем производится запись строки в порт. Обмен с портом записывается в файл.

```
Порт    = СОМОТКРЫТЬ( "COM7,9600,8,N,O,N", "d:\comlog.txt" );
СООБЩЕНИЕ( СтрокаИзМассива( СОМПРОЧИТАТЬ( Порт, -1, 10000 ) ) );
СООБЩЕНИЕ( СОМЗАПИСАТЬ( Порт, МАССИВИЗСТРОКИ( "COM7,9600,8,N,O,N" ), 1000 ) );
СОМЗАКРЫТЬ( Порт );
```

ФУНКЦИИ ДЛЯ РАБОТЫ С **ФАЙЛАМИ**

ФАЙЛСОЗДАТЬ(ИмяФайла[, Флаги])
FILECREATE(ИмяФайла[, Флаги])
Создает файл с указанным именем в указанном каталоге и возвращает дескриптор созданного файла в случае успеха или -1, если не удалось создать указанный файл.
Параметры.
ИмяФайла- Строка, имя создаваемого файла, должна содержать полный путь к создаваемому файлу.
Флаги- Число, варианты создания файла.
Допустимые значения для данного параметра:
СОЗДАВАТЬ ВСЕГДА 0,
СОЗДАТЬ НОВЫЙ 1.
По умолчанию используется значение **СОЗДАВАТЬ ВСЕГДА.**
Возвращаемое значение - Число целое положительное, дескриптор созданного файла, который должен использоваться во всех функциях для работы с файлом. В случае возникновения ошибки функция возвращает значение -1.

ФАЙЛОТКРЫТЬ(ИмяФайла[, Флаги[, Доступ]])
FILEOPEN(ИмяФайла[, Флаги[, Доступ]])
Открывает файл с указанным именем и возвращает дескриптор открытого файла в случае успеха или -1, если не удалось открыть указанный файл.
Параметры.
ИмяФайла- Строка, имя открываемого файла, должна содержать полный путь к открываемому файлу.
Флаги- Число, варианты открытия файла.
Допустимые значения для данного параметра:
ОТКРЫТЬ СУЩЕСТВУЮЩИЙ 0,
ОТКРЫВАТЬ ВСЕГДА 1,
ЗАПРОСИТЬ ИМЯ ФАЙЛА 2.
По умолчанию используется значение **ОТКРЫТЬ СУЩЕСТВУЮЩИЙ.**
Доступ- Число, варианты требуемого доступа к файлу.
Допустимые значения для данного параметра:
ТОЛЬКО ДЛЯ ЧТЕНИЯ 0,
ДЛЯ ЧТЕНИЯ И ЗАПИСИ 1,
ТОЛЬКО ДЛЯ ЗАПИСИ 2.
По умолчанию используется значение **ТОЛЬКО ДЛЯ ЧТЕНИЯ.**
Возвращаемое значение - Число целое положительное, дескриптор открытого файла, который должен использоваться во всех функциях для работы с файлом. В случае возникновения ошибки функция возвращает значение -1.

ФАЙЛЗАКРЫТЬ(ДескрипторФайла)
FILECLOSE(ДескрипторФайла)
Закрывает ранее открытый или созданный файл.
Параметры.
ДескрипторФайла- Число целое, значение возвращенное функциями создания или открытия файла.
Возвращаемое значение - Логическое. Истина, если файл был успешно закрыт. Ложь, если был передан неверный дескриптор файла.

ФАЙЛУДАЛИТЬ(ИмяФайла)
FILEDELETE(ИмяФайла)

Удаляет указанный файл.
Параметры.
ИмяФайла- Строка, имя существующего файла, который необходимо удалить.
Возвращаемое значение - Число целое равное количеству удаленных файлов.

ФАЙЛПЕРЕИМЕНОВАТЬ(ИмяИсходногоФайла, ИмяРезультирующегоФайла)
FILERENAME(ИмяИсходногоФайла, ИмяРезультирующегоФайла)
Функция переименовывает или переносит исходный файл в результирующий.
Параметры.
ИмяИсходногоФайла- Строка, имя существующего файла, который необходимо переименовать или переместить.
ИмяРезультирующегоФайла- Строка, имя файла, в который необходимо переименовать или переместить исходный файл.
Возвращаемое значение - Число целое равное количеству переименованных и перенесенных файлов.

ФАЙЛПРОЧИТАТЬ(ДескрипторФайла[,ТипЗначения[,КоличествоБайт[,ОграничительСтрок]]])
FILEREAD(ДескрипторФайла[,ТипЗначения[,КоличествоБайт[,ОграничительСтрок]]])
Читает указанный блок данных из файла.
Параметры.
ДескрипторФайла- Число целое, значение возвращенное функциями создания или открытия файла.
ТипДанных- Строка состоящая из одного символа, определяющего тип возвращаемых функцией данных. Допустимые значения для данного параметра:
S- строка,
I- число целое,
F- число вещественное,
D- дата и время,
L- логическое,
B- двоичные данные.
По умолчанию используется значение S.
КоличествоБайт- Число целое. Может использоваться для указания количества читаемых данных для типов S и B. Если данный параметр не указан, то для типа S будут прочитаны данные до первых символов указанных в параметре **ОграничительСтрок**, для типа данных B файл будет прочитан до конца.
ОграничительСтрок- Строка, символы которой являются ограничителями строк. Если параметр не указан, то ограничителями строк являются символы CR (возврат каретки) и LF (перевод строки).
Возвращаемое значение - зависит от типа затребованных данных. В случае ошибки возвращается пустое значение затребованного типа.

ФАЙЛНАЙТИ(ШаблонИмениФайла[, ИскатьПодкаталоги])
FILEFINDFIRST(ШаблонИмениФайла[, ИскатьПодкаталоги])
Функция ищет первый файл, имя которого удовлетворяет указанному шаблону. Также возможен поиск подкаталога, имя которого удовлетворяет шаблону.
Параметры.
ШаблонИмениФайла- Строка, шаблон имени файл для поиска. Шаблон может содержать символы * и ?.
ИскатьПодкаталоги- Логическое, указывает что необходимо искать подкаталоги, имена которых удовлетворяют указанному шаблону.
Возвращаемое значение - Строка, содержащая полное имя найденного файла и подкаталога.

ФАЙЛНАЙТИСЛЕДУЮЩИЙ([ИскатьПодкаталоги])
FILEFINDNEXT([ИскатьПодкаталоги])
Функция ищет следующий файл, имя которого удовлетворяет указанному шаблону. Шаблон задается в предшествующем вызове функции НАЙТИФАЙЛ. Также возможен поиск подкаталога, имя которого удовлетворяет шаблону.
Параметры.

ИскатьПодкаталоги- Логическое, указывает что необходимо искать подкаталоги, имена которых удовлетворяют указанному шаблону.
Возвращаемое значение - Строка, содержащая полное имя найденного файла и подкаталога.

ФАЙЛЗАКОНЧИТЬПОИСК()
FILEFINDCLOSE()
Функция завершает поиск файлов начатый функцией НАЙТИФАЙЛ и освобождает системные ресурсы связанные с поиском файлов.
Параметры.
нет.
Возвращаемое значение - Логическое. Истина, в случае успешного завершения, иначе Ложь.

ФАЙЛЗАПИСАТЬ(ДескрипторФайла|ИмяФайла, Значение[,КоличествоБайт])
FILEWRITE(ДескрипторФайла|ИмяФайла, Значение[,КоличествоБайт])
Записывает указанный блок данных в файл.
Параметры.
ДескрипторФайла- Число целое, значение возвращенное функциями создания или открытия файла.
ИмяФайла- Строка, имя файла, в который необходимо записать данные. Если указано имя файла, то файл будет открыт на запись и сразу закрыт после окончания записи данных.
Значение- Значение любого типа, которое будет целиком или частично записано в файл.
КоличествоБайт- Число целое. Может использоваться для указания количества записываемых байт данных. Если данный параметр не указан, то Значение будет записано в файл целиком.
Возвращаемое значение - Число целое равное количеству записанных байт данных.
В случае ошибки возвращается значение ноль.

ФАЙЛЗАПИСАТЬСТРОКУ(ДескрипторФайла|ИмяФайла, Строка)
FILEWRITESTRING(ДескрипторФайла|ИмяФайла, Строка)
Записывает указанную строку в файл. После строки записываются два управляющих символа - Возврат каретки (CR) и Перевод строки (LF).
Параметры.
ДескрипторФайла- Число целое, значение возвращенное функциями создания или открытия файла.
ИмяФайла- Строка, имя файла, в который необходимо записать данные. Если указано имя файла, то файл будет открыт на запись и сразу закрыт после окончания записи данных.

Строка- Строка, которая будет целиком записана в файл.
Возвращаемое значение - Логическое. Истина, если запись прошла успешно, иначе Ложь.

ФАЙЛУСТАНОВИТЬУКАЗАТЕЛЬ(ДескрипторФайла, МладшиеРазрядыУказателя[, СтаршиеРазрядыУказателя[, МетодПеремещения]])
FILESETPOINTER(ДескрипторФайла, МладшиеРазрядыУказателя[, СтаршиеРазрядыУказателя[, МетодПеремещения]])
Перемещает указатель в указанном открытом файле на указанную позицию. Позиция может быть как 32-х битным целым числом, так и 64-х битным целым числом.
Параметры.
ДескрипторФайла- Число целое, значение возвращенное функциями создания или открытия файла.
МладшиеРазрядыУказателя- Число целое являющееся младшими 32-мя битами указателя в файле.

СтаршиеРазрядыУказателя- Число целое являющееся старшими 32-мя разрядами 64-х битного указателя в файле.
МетодПеремещения- Число целое, указывающее на метод перемещения указателя. Допустимые значения для данного параметра:
от начала файла- 0,
от текущей позиции- 1,
от конца файла- 2.
По умолчанию используется значение 0.
Возвращаемое значение - Логическое. Истина, в случае успешного перемещения указателя. Ложь - в случае ошибки.

ФАЙЛУКАЗАТЕЛЬ(ДескрипторФайла[, СтаршиеРазряды])
FILEGETPOINTER(ДескрипторФайла[, СтаршиеРазряды])
Возвращает значение указателя в указанном открытом файле. Позиция может быть как 32-х битным целым числом, так и 64-х битным целым числом.
Параметры.
ДескрипторФайла- Число целое, значение возвращенное функциями создания или открытия файла.
СтаршиеРазряды- Логическое. Если параметр равен Истине, то функция возвращает старшие 32 разряда 64-х битного указателя в файле, иначе возвращаются младшие 32 разряда 64-х битного указателя в файле. Если параметр не указан, то используется значение Ложь.
Возвращаемое значение - Число целое равное либо младшим, либо старшим разрядам 64-х битного указателя в указанном файле.

ФАЙЛРАЗМЕР(ДескрипторФайла[, СтаршиеРазряды])
FILEGETSIZE(ДескрипторФайла[, СтаршиеРазряды])
Возвращает размер указанного файла. Размер файла может быть как 32-х битным целым числом, так и 64-х битным целым числом.
Параметры.
ДескрипторФайла- Число целое, значение возвращенное функциями создания или открытия файла.
СтаршиеРазряды- Логическое. Если параметр равен Истине, то функция возвращает старшие 32 разряда 64-х битного размера файла, иначе возвращаются младшие 32 разряда 64-х битного размера файла. Если параметр не указан, то используется значение Ложь.
Возвращаемое значение - Число целое равное либо младшим, либо старшим разрядам 64-х битного размера файла.

ФАЙЛКОНЕЦФАЙЛА(ДескрипторФайла)
FILEEOF(ДескрипторФайла)
Функция определяет, находится ли указатель файла за пределами конца файла.
Параметры.
ДескрипторФайла- Число целое, значение возвращенное функциями создания или открытия файла.
Возвращаемое значение - Логическое. Истина, если указатель находится за пределами конца файла. Иначе Ложь.

ФАЙЛКОЛИЧЕСТВОСТРОК(ДескрипторФайла[, ОграничительСтрок])
FILELINESCOUNT(ДескрипторФайла[, ОграничительСтрок])
Функция определяет количество строк текста в указанном файле. Строки делятся последовательностью символов ОграничительСтрок
Параметры.
ДескрипторФайла- Число целое, значение возвращенное функциями создания или открытия файла.
ОграничительСтрок- Строка, символы которой являются ограничителями строк. Если параметр не указан, то ограничителями строк являются символы CR (возврат каретки) и LF (перевод строки)

Возвращаемое значение - Число целое равное количеству строк текста в указанном файле.

ФАЙЛСПИСОК(ШаблонИмёнФайлов[, ИмяМассива[, Флаги]])
FILESLIST(ШаблонИмёнФайлов[, ИмяМассива[, Флаги]])
Функция копирует имена файлов, удовлетворяющие указанному шаблону, в массив переменных. Если имя массива не указано, то будет создана временная таблица в текущем источнике данных со следующей структурой:

Filename	- Символьное, имя файла без полного пути.
Pathname	- Имя файла с полным путём к нему.
Filesize	- Число, размер файла в байтах.
Folder	- Логическое, ИСТИНА, если строка описывает папку.
Created	- Дата и время создания файла.
Lastmodified	- Дата и время последнего изменения файла.
Attr	- Строка, атрибуты файла.

Таблица будет создана с уникальным именем, и в нее будет произведена запись информации о найденных файлах.
Параметры.
ШаблонИмёнФайлов- Строка содержащая шаблон имён файлов для поиска и копирования.
ИмяМассива- Строка, имя массива, который будет создан и заполнен именами файлов.
Флаги- Число целое,
1 - Заполнять список только папками,
0 - только файлами.
Если параметр не указан, то используется значение 0.
Возвращаемое значение - Число целое равное количеству найденных файлов, если запись производится в массив, или имя временной таблицы, если запись была произведена во временную таблицу.

ФАЙЛ(ШаблонИмениФайла[, Флаги])
FILEEXISTS(ШаблонИмениФайла[, Флаги])
Функция определяет, существует ли файл с именем удовлетворяющим указанному шаблону.
Параметры.
ШаблонИмениФайла- Строка содержащая шаблон имени файла для поиска.
Флаги- Число целое, 1 - Искать только папки, 0 - искать только файлы. Если параметр не указан, то используется значение 0.
Возвращаемое значение - логическое. ИСТИНА, если указанный файл существует. Иначе - ЛОЖЬ.

ФАЙЛУПАКОВАТЬ(ИмяФайла[, ИмяУпакованногоФайла])
FILEPACK(ИмяФайла[, ИмяУпакованногоФайла])
Функция упаковывает (сжимает) указанный файл ИмяФайла.
Параметры.
ИмяФайла - Строка содержащая имя файла для упаковки.
ИмяУпакованногоФайла - Строка, имя упакованного файла. Если не указано, то используется имя исходного файла и добавляется расширение ZIP.
Возвращаемое значение - логическое. **ИСТИНА**, если упаковка проведена успешно. Иначе - **ЛОЖЬ**.

ФАЙЛРАСПАКОВАТЬ(ИмяУпакованногоФайла, ИмяФайла)
FILEUNPACK(ИмяУпакованногоФайла, ИмяФайла)
Функция распаковывает файл ИмяУпакованногоФайла в файл с именем ИмяФайла.
Параметры.
ИмяУпакованногоФайла - Строка содержащая имя файла для распаковки.
ИмяФайла - Строка, имя распакованного файла.
Возвращаемое значение - логическое. **ИСТИНА**, если распаковка проведена успешно. Иначе - **ЛОЖЬ**.

ФАЙЛВЫБРАТЬ(ЧтениеЗапись,Фильтр[,Флаги[,РасширениеПоУмолчанию[,ИмяФайла]]]

FILESELECT(ЧтениеЗапись,Фильтр[,Флаги[,РасширениеПоУмолчанию[,ИмяФайла]]])
Функция позволяет выбрать файл, который необходимо будет прочитать или записать. Функция производит вызов системной функции GetOpenFileName (GetSaveFileName).
Параметры.
ЧтениеЗапись – Логическое. **ИСТИНА**, если необходимо отобразить окно для открытия файла. **ЛОЖЬ**, если для записи файла.
Фильтр – Строка, содержащая список фильтров для отображения имен файлов. Каждый фильтр записывается в форме **НазваниеФильтра|ШаблонИменФайлов**. Например. "Текстовые файлы|*.txt".
Флаги – Число целое, флаги управляющие поведением окна выбора. Флаги должны являться комбинацией одного или нескольких значений:

- 1 – при открытии окна будет установлен признак **Только для чтения**
- 2 – при перезаписи существующего файла будет предупреждение
- 4 – не выводить признак Только для чтения
- 8 – после выбора файла возвращать рабочий
- 512 – позволить выбор нескольких файлов
- 2048 – не разрешать вводить неправильный путь в строку имени файла
- 4096 – проверять наличие файла, имя которого вводит пользователь
- 131072 – убрать кнопку Сеть
- 524288 – разрешить изменение размеров окна

РасширениеПоУмолчанию – Строка, расширение, которое будет добавлено к имени файла, если пользователь его не указал.
ИмяФайла – Строка, имя файла, которое будет отображено в строке имени, при открытии окна.
Возвращаемое значение – Строка, имя выбранного файла с полным путем. Пустая строка, если пользователь отказался от выбора файла.

ФАЙЛЗАПИСАТЬВЖУРНАЛ(ИмяФайла,Сообщение)
FILEWRITELOG(ИмяФайла,Сообщение[,ЗаписыватьВремя])
Функция открывает существующий или создает новый файл и дописывает в него переданное сообщение.
Параметры.
ИмяФайла– Строка, имя файла, в который будет записано сообщение.
Сообщение– Строка, которая будет добавлена в файл журнала.
ЗаписыватьВремя– Логическое, если **ИСТИНА**, то в начале строки будет записана текущая дата и время. Если параметр не указан, то используется значение ИСТИНА.
Возвращаемое значение – Логическое, **ИСТИНА**, если запись была успешно добавлена. Иначе **ЛОЖЬ**.

КАТАЛОГСОЗДАТЬ(ИмяКаталога)
DIRECTORYCREATE(ИмяКаталога)
Создает каталог с указанным именем по указанному пути.
Параметры.
ИмяКаталога– Строка, имя создаваемого каталога, должна содержать полный путь к создаваемому каталогу. Путь должен существовать заранее.
Возвращаемое значение – Логическое, **ИСТИНА**, если каталог был создан. Иначе, **ЛОЖЬ**.

ФУНКЦИИ ДЛЯ ФОРМИРОВАНИЯ XML ТЕКСТОВ.

Следующие функции позволяют формировать XML тексты последовательно добавляя в текст новые тэги. Текст может автоматически конвертироваться в кодировку utf-8. Если предполагается формирование большого объема текста, то можно сразу указать на необходимость записи текста в файл, минуя формирование в памяти. Формирование XML начинается с создания объекта XML функцией **XMLНАЧАТЬ** / **XMLSTARTDOCUMENT**. Завершается обязательным вызовом функции

XMLЗАКОНЧИТЬ / **XMLENDDOCUMENT**. Функции **XMLЗАКОНЧИТЬ** / **XMLENDDOCUMENT** освобождают занятые ресурсы, закрывают файлы и удаляют объект XML.

XMLНАЧАТЬ([ИмяФайла[, Кодировка]])
XMLSTARTDOCUMENT([ИмяФайла[, Кодировка]])
Функция создает объект для формирования XML текста. Внутренний объект будет поддерживать стек имен открытых тэгов, преобразование кодировок и, при необходимости, прямую запись в файл. Функция возвращает дескриптор объекта. Этот дескриптор необходимо будет указывать в вызовах всех последующих функций, формирующих XML текст.
Параметры.
ИмяФайла – Строка, если строка не пустая, то XML текст будет записываться в указанный файл. Если имя файла не указано, или указано пустое, то XML текст будет формироваться во внутреннем буфере. Результат форматирования будет возвращен функцией **XMLЗАКОНЧИТЬ** или **XMLENDDOCUMENT**. Если будет указано неверное имя файла, т.е. не удастся создать файл с указанным именем, то функция не перейдет в режим формирования во внутреннем буфере, а возвращенный объект будет содержать ошибку создания файла (код ошибки 2).
Кодировка – Строка, указывающая, в какой кодировке должен формироваться XML текст. Может принимать значения ANSI или UTF-8. Если выбран вариант ANSI, то действительная кодировка будет зависеть от текущей кодовой страницы **Windows**. Если параметр не указан или передано неверное/пустое значение, то будет использоваться кодировка utf-8.
Возвращаемое значение – Число, дескриптор объекта. Если будет возвращено -1, то значит произошла ошибка.

XMLЗАПИСАТЬТЕГ(ДескрипторXML, ИмяТега[, Атрибуты[, Текст[, ФорматТекста[, ПространстваИмен]]]])
XMLWRITEELEMENT(ДескрипторXML, ИмяТега[, Атрибуты[, Текст[, ФорматТекста[, ПространстваИмен]]]])
Функция записывает указанный тег в выходной поток объекта XML. Тег может содержать атрибуты, текст и дополнительные определения пространств имен.
Параметры.

ДескрипторXML – Число, дескриптор объекта XML, возвращенный функцией XMLНАЧАТЬ.

ИмяТега – Строка, имя тега, возможно с префиксом пространства имен.

Атрибуты – Строка, список через запятую, с названиями атрибутов. Названия атрибутов должны совпадать с именами переменных, которые содержат значения для атрибутов. Имени атрибута может предшествовать префикс пространства имен отделенный двоеточием. В XML будет подставлена строка вида ns:ИмяПеременной="Значение". Если переменной с указанным именем нет, то атрибут будет пропущен.

Текст – Строка, текст, записываемый в тело тега.

Формат – Строка, допустимые значения:

- BASE64 – текст будет преобразован в кодировку BASE64 и записан в тело тега. Полезно, если текст содержит запрещенные для XML символы.
- CDATA – Тело тега будет добавлено в служебное слово CDATA, позволяющее экранировать произвольный текст.
- PLAIN, или пусто (ничего не указано) – текст будет записан «как есть».

ПространстваИмен – Строка, список через запятую пространств имен. Аналогично атрибутам, строка должна содержать имена переменных, совпадающих с префиксами пространств имен. В текст будут добавлены строки вида *xmlns:ИмяПеременной="Значение"*.

Возвращаемое значение – Число целое, код текущей ошибки. Если ошибок нет, то 0.

XMLНАЧАЛОТЕГА(ДескрипторXML, ИмяТега[,Атрибуты[,ПространстваИмен]])
XMLWRITEELEMENTSTART(ДескрипторXML, ИмяТега[, Атрибуты[, ПространстваИмен]])
Функция записывает начало тега в выходной поток объекта XML. Тег может содержать атрибуты и дополнительные определения пространств имен. Под началом тега подразумевается конструкция вида *<ИмяТега xmlns:ns= "Uri" Атрибут="Значение" >*, где ns – префикс пространства имен. Пространств имен и атрибутов может быть несколько.

Параметры.

ДескрипторXML Число, дескриптор объекта XML, возвращенный функцией XMLНАЧАТЬ.

ИмяТега Строка, имя тега, возможно с префиксом пространства имен.

Атрибуты Строка, список через запятую, с названиями атрибутов. Названия атрибутов должны совпадать с именами переменных, которые содержат значения для атрибутов. Имени атрибута может предшествовать префикс пространства имен отделенный двоеточием. В XML будет подставлена строка вида ns:ИмяПеременной="Значение". Если переменной с указанным именем нет, то атрибут будет пропущен.

ПространстваИмен Строка, список через запятую пространств имен. Аналогично атрибутам, строка должна содержать имена переменных, совпадающих с префиксами пространств имен. В текст будут добавлены строки вида *xmlns:ИмяПеременной="Значение"*.

Возвращаемое значение – Число целое, код текущей ошибки. Если ошибок нет, то 0.

XMLТЕКСТТЕГА(ДескрипторXML, Текст[, ФорматТекста])
XMLWRITEELEMENTTEXT(ДескрипторXML, Текст[, ФорматТекста])
Функция записывает в текущее расположение текст тега.

Параметры.

ДескрипторXML Число, дескриптор объекта XML, возвращенный функцией **XMLНАЧАТЬ.**

Текст Строка, текст, записываемый в тело тега.

Формат Строка, допустимые значения:

- BASE64 – текст будет преобразован в кодировку BASE64 и записан в тело тега. Полезно, если текст содержит запрещенные для XML символы.
- CDATA – Тело тега будет добавлено служебное слово CDATA, позволяющее экранировать произвольный текст.
- PLAIN, или пусто (ничего не указано) – текст будет записан «как есть».

Возвращаемое значение – Число целое, код текущей ошибки. Если ошибок нет, то 0.

XMLКОММЕНТАРИЙ(ДескрипторXML, Комментарий)
XMLWRITECOMMENT(ДескрипторXML, Текст[, ФорматТекста])
Функция записывает в текущее расположение комментарий к XML тексту.

Параметры.

ДескрипторXML Число, дескриптор объекта XML, возвращенный функцией **XMLНАЧАТЬ.**

Комментарий Строка, комментарий, записываемый в тело тега.

Возвращаемое значение – Число целое, код текущей ошибки. Если ошибок нет, то 0.

XMLЗАВЕРШИТЬТЕГ(ДескрипторXML)
XMLWRITEELEMENTEND(ДескрипторXML)
Функция записывает окончание открытого ранее функцией **XMLНАЧАЛОТЕГА** / **XMLWRITEELEMENTSTART** тега в выходной поток объекта XML. Запись начала тэга помещает на стек записываемый тег. Функция завершения тега берет со стека значение и записывает строку вида *</ns:ИмяТега>* в выходной поток объекта. Если

была нарушена последовательность вызова функций и на стеке ничего нет, то функция ничего не делает, а устанавливает код ошибки 6 (нарушена последовательность вызовов функций).
Параметры.
ДескрипторXML– Число, дескриптор объекта XML, возвращенный функцией XMLНАЧАТЬ.
Возвращаемое значение – Число целое, код текущей ошибки. Если ошибок нет, то 0.

XMLКОДОШИБКИ(ДескрипторXML)
XMLERRORCODE(ДескрипторXML)
Функция возвращает текущий код ошибки, возникшей после последней или более ранней выполненной операции с объектом XML.
Коды ошибок и их расшифровка

0	Нет ошибок
1	Ошибка кодирования. Текст содержит запрещенные форматом XML символы. Ошибочные символы не записываются в выходной поток, а пропускаются.
2	Не удалось создать файл с указанным именем. Недостаточно прав или неверно указана папка или имя файла. Объект переводится в режим формирования текста во внутреннем буфере.
3	Ошибка записи файла. Если файл был создан, то вероятнее всего закончилось место на диске. Критическая ошибка, препятствующая дальнейшему формированию текста.
4	Недостаточно памяти. Ошибка может возникнуть, если текст формируется в памяти и стал чрезмерно большим.
5	Переданы неверные параметры функции.
6	Неверная последовательность вызова функций, приведшая к тому, что стек тегов стал пустым.

Параметры.
ДескрипторXML– Число, дескриптор объекта XML, возвращенный функцией **XMLНАЧАТЬ**.
Возвращаемое значение – Число, код ошибки. 0, если ошибок не было.

XMLОЧИСТИТЬОШИБКИ(ДескрипторXML)
XMLCLEARERROR(ДескрипторXML)
Функция очищает текущий код ошибки, возникшей после последней или более ранней выполненной операции с объектом XML.
Параметры.
ДескрипторXML– Число, дескриптор объекта XML, возвращенный функцией **XMLНАЧАТЬ**.
Возвращаемое значение – функция не возвращает результат.

XMLЗАКОНЧИТЬ(ДескрипторXML)
XMLENDDOCUMENT(ДескрипторXML)
Функция завершает формирование XML и возвращает полученный результат. Если запись производилась в файл, то возвращается имя этого файла. Иначе, возвращает сформированный XML текст. Если на момент вызова функции в стеке находятся незакрытые теги, то все теги закрываются. Если запись производилась в файл, то файл закрывается. Объект удаляется. Дальнейшая запись в него становится невозможной.
Параметры.
ДескрипторXML– Число, дескриптор объекта XML, возвращенный функцией **XMLНАЧАТЬ**.
Возвращаемое значение – Строка, либо имя файла, либо текст XML.

ФУНКЦИИ ДЛЯ РАБОТЫ С ZIP АРХИВАМИ

АРХИВОТКРЫТЬ(ИмяФайла | ДвоичныеДанные)
ZIPOPEN(ИмяФайла | ДвоичныеДанные)
Открывает архив для дальнейшей работы. Архив может быть файлом или двоичными данными, например, полученными по каналам связи. В случае указания имени файла будет произведена проверка наличия такого файла. Если файл существует, то он будет открыт и проверен на формат zip архива. Если существующий файл не является

zip архивом, то функция вернет ошибку. Двоичные данные должны представлять собой zip архив, только не записанный на носитель. В случае, если переданный массив не является правильным архивом – будет возвращена ошибка. Значение, возвращаемое при успешном открытии архива необходимо использовать в последующих вызовах функций для работы с архивом.
Предварительное открытие архива не является обязательным. Оно удобно для выполнения последовательности дальнейших операций с архивом, с возможностью отказаться от внесенных изменений.
После окончания работы с архивом, необходимо его закрыть функцией **АРХИВЗАКРЫТЬ( ) / ZIPCLOSE( ).**
Параметры.
ИмяФайла- Строка с именем файла архива.
ДвоичныеДанные- Двоичный массив, содержащий архив.
Возвращаемое значение – число целое. Если больше или равно нулю, то номер архива. -1 в случае возникновения ошибки. Код и описание причины ошибки возвращается в переменных **_ERRORCODE** и **_ERRORDESCRIPTION.**

АРХИВЗАКРЫТЬ(НомерАрхива[, ОтменитьИзменения])
ZIPCLOSE(НомерАрхива[, ОтменитьИзменения])
Закрывает ранее открытый архив. В момент закрытия, если не указано, что нужно отменить изменения, производится фактическая запись архива на носитель или в память, если архив был открыт в памяти.
Параметры.
НомерАрхива- Число целое. Значение, возвращенное функцией **АРХИВОТКРЫТЬ( ) / ZIPOPEN( ).**
ОтменитьИзменения- Логическое. Если **Истина**, то архив будет закрыт, а внесенные изменения отменены. По умолчанию используется значение **Ложь.**
Возвращаемое значение – Логическое или двоичный массив. Если архив был открыт из файла, то возвращается логическое значение. **Истина**, при успешной записи на диск, иначе, **Ложь.** Код и описание причины ошибки возвращается в переменных **_ERRORCODE** и **_ERRORDESCRIPTION.** Если архив был открыт в памяти, то возвращается двоичный массив.

АРХИВ(Номер | ИмяАрхива | ДвоичныйАрхив, Операция[, ИмяФайлаВАрхиве | НомерФайлаВАрхиве [, ИмяФайлаВОС | МассивДанных[, Комментарий[, Флаги]]]])
ZIP(Номер | ИмяАрхива | ДвоичныйАрхив, Операция[, ИмяФайлаВАрхиве | НомерФайлаВАрхиве [, ИмяФайлаВОС | МассивДанных[, Комментарий[, Флаги]]]])
Выполняет указанную операцию с архивом. Операция может быть выполнена как с ранее открытым архивом, как часть пакета операций, так и как одиночная операция с единичным архивом. Во втором случае, функция автоматически откроет указанный архив, выполнит операцию и закроет архив. В зависимости от параметров, функция может выполнять операции с файловыми архивами и архивами в памяти. В различных комбинациях. Функция предполагает, что все имена файлов в архиве записаны в кодировке **UTF-8.** Все значения параметров автоматически преобразуются к этой кодировке. На вход функции они должны передаваться в кодировке **ANSI.**
Параметры.
НомерАрхива- Число целое. Значение, возвращенное функцией **АРХИВОТКРЫТЬ( ) / ZIPOPEN( ).**
ИмяАрхива- Строка. Имя файла с архивом.
ДвоичныйАрхив- Двоичный массив данных, содержащий архив. Например, данные полученные по каналам связи.

Операция- Строка, выполняемая операция. Допустимые значения:

- **ADD | ДОБАВИТЬ** – Добавить файл в архив. Используются параметры: **ИмяФайлаВАрхиве, ИмяФайлаВОС | МассивДАнных, Флаги.** Добавляет единичный файл или группу файлов в архив. Если в параметре **ИмяФайлаВОС** указан отдельный файл, то этот файл будет добавлен в архив. Если **ИмяФайлаВОС** указывает на папку, то будут добавлены все файлы из папки. Параметр **ИмяФайлаВОС** может содержать символы маски *****

и **?**. В этом случае, будут обработаны все файлы и папки удовлетворяющие указанной маске. Например, Если параметр **ИмяФайлаВОС** равен **"D:\t*mp*.xml"**, то в архив будут добавлены **XML** файлы из всех папок, подходящих под маску **t*mp** (temp, tamp, t123mp и т.д.). Если вместо **ИмяФайлаВОС** будет задано **МассивДанных**, то функция добавит одиночный файл из массива двоичных данных. Файл в архиве будет иметь имя **ИмяФайлаВАрхиве**. Если **ИмяФайлаВОС** не будет задано, а параметр **ИмяФайлаВАрхиве** будет заканчиваться косой чертой (/ или \), то в архив будет добавлена запись о каталоге. *Примечание. Для добавления каталогов в архив параметр **Флаги** должен содержать значение **КЕЕРРАТН | СОХРАНЯТЬПУТЬ**.* Если параметр ИмяФайлаВОС указывает на папку с файлами, то для того, чтобы в архив были добавлены все вложенные папки, Флаги должен содержать **RECURSION | РЕКУРСИЯ**. Если в процессе добавления файлов в архив какой-то файл недоступен для чтения, например, занят другой программой, то функция может либо остановиться с ошибкой, либо пропустить недоступный файл. Поведение регулируется флагом **SKIPINACCESSIBLE | ПРОПУСКАТЬНЕДОСТУПНЫЕ**.

- **FILESLIST | СПИСОКФАЙЛОВ –** Формирование массива со списком имен файлов в архиве. Имя формируемого массива передается в параметре **ИмяМассива**. Будет возвращено целое число, равное количеству отдельных записей файлов в архиве. Это могут быть как файлы, так и отдельные записи о каталогах. Если массив уже существует, то он будет предварительно очищен.
- **COUNT | КОЛИЧЕСТВОФАЙЛОВ –** Подсчет количества файлов в архиве. Будет возвращено целое число, равное количеству отдельных записей файлов в архиве. Это могут быть как файлы, так и отдельные записи о каталогах.
- **FILENAME | ИМЯФАЙЛА** – возвращает имя файла из указанного номера записи в архиве. Используются параметры: **НомерФайлаВАрхиве**. Если полученное имя будет заканчиваться косой чертой (/ или \), то это значит, что в запрошенной позиции не файл а имя каталога.
- **FILESIZE | РАЗМЕРФАЙЛА** – возвращает размер файла из указанного номера записи в архиве. Используются параметры: **НомерФайлаВАрхиве**. Для записей каталогов возвращается 0.
- **FILEINDEX | НОМЕРФАЙЛА** – возвращает номер записи файла в архиве по его имени. Используются параметры: **ИмяФайлаВАрхиве**, **Флаги**. Если параметр Флаги содержит флаг **NOCASE | БЕЗРЕГИСТРА**, то поиск производится без учета регистра букв. Если файл с указанным именем не найден в архиве, то будет возвращено значение -1.
- **REMOVE | УДАЛИТЬ** – удаляет указанный файл из архива. Используются параметры: **ИмяФайлаВАрхиве | НомерФайлаВАрхиве**, **Флаги**. Если параметр **Флаги** содержит флаг **NOCASE | БЕЗРЕГИСТРА**, то поиск удаляемого файла производится без учета регистра букв. Если указан **НомерФайлаВАрхиве**, то параметр **Флаги** игнорируется.
- **EXCTRACT | ИЗВЛЕЧЬ** – извлечь файл из архива. Используются параметры: **ИмяФайлаВАрхиве | НомерФайлаВАрхиве**, **ИмяФайлаВОС**, **Флаги**. Если указано **ИмяФайлаВАрхиве**, не содержащее макросимволов (***** и/или **?**) или задан **НомерФайлаВАрхиве**, то функция извлечет только один указанный файл. Если **ИмяФайлаВАрхиве** или **НомерФайлаВАрхиве** указывают на каталог, а **Флаги** не содержат **CREATEFOLDERS | СОЗДАВАТЬКАТАЛОГИ**, то функция ничего не сделает. Если **Флаги** содержат **CREATEFOLDERS | СОЗДАВАТЬКАТАЛОГИ**, то функция создаст необходимые папки. Если параметр **ИмяФайлаВАрхиве** содержит макросимволы (***** и/или **?**), то будут извлечены все файлы, удовлетворяющие маске. При извлечении файлов могут быть созданы все необходимые каталоги. Для этого параметр **Флаги** должен содержать **CREATEFOLDERS | СОЗДАВАТЬКАТАЛОГИ**. Параметр ИмяФайлаВОС указывает куда записывать результат.
    - Если **ИмяФайлаВОС** пустое, то результат будет возвращен в виде двоичного массива, а не записан в файл. При этом, если

ИмяФайлаЫАрхиве содержит маску, то будет извлечен только первый подходящий файл.

- Если **ИмяФайлаВОС** не пустое, содержит полный путь и имя файла, т.е. заканчивается не именем существующей папки, то если:
  - **ИмяФайлаВАрхиве** указывает на один файл, или задан **НомерФайлаВАрхиве**, то извлекаемый файл будет записан на диск с именем **ИмяФайлаВОС**.
  - **ИмяФайлаВАрхиве** содержит макро символы (***** и/или **?**), то это будет считаться ошибочной ситуацией.
- Если **ИмяФайлаВОС** содержит путь к папке без имени файла, то извлекаемые файлы будут помещены в эту папку. При этом, если **Флаги** содержит **CREATEFOLDERS | СОЗДАВАТЬКАТАЛОГИ**, то будут созданы все необходимые вложенные папки, содержащиеся в именах извлекаемых файлов.

- **ADDCOMMENT | ДОБАВИТЬКОММЕНТАРИЙ** – позволяет добавить комментарий к файлу в архиве или к самому архиву. Используются параметры: **ИмяФайлаВАрхиве | НомерФайлаВАрхиве**, **Комментарий**, **Флаги**. **ИмяФайлаВАрхиве** должно указывать на единичный файл. Если параметр **Флаги** содержит флаг **NOCASE | БЕЗРЕГИСТРА**, то поиск файла производится без учета регистра букв. Если указан **НомерФайлаВАрхиве**, то параметр **Флаги** игнорируется. Если не указано ни **ИмяФайлаВАрхиве**, ни **НомерФайлаВАрхиве**, то комментарий будет прикреплен к самому архиву. Если у файла / архива уже есть комментарий, то он будет заменен на новый.
- **COMMENT | КОММЕНТАРИЙ** – позволяет получить комментарий к файлу в архиве или к самому архиву. Используются параметры: **ИмяФайлаВАрхиве | НомерФайлаВАрхиве**, **Флаги**. **ИмяФайлаВАрхиве** должно указывать на единичный файл. Если параметр **Флаги** содержит флаг **NOCASE | БЕЗРЕГИСТРА**, то поиск файла производится без учета регистра букв. Если указан **НомерФайлаВАрхиве**, то параметр **Флаги** игнорируется. Если не указано ни **ИмяФайлаВАрхиве**, ни **НомерФайлаВАрхиве**, то возвращается комментарий к самому архиву.
- **TEST | ПРОВЕРИТЬ** – выполняет проверку архива. Проверка производится методом чтения всех файлов в архиве без реальной записи на диск. Возвращается значение Истина или Ложь, в зависимости от результата проверки.

ИмяФайлаВАрхиве- Строка. При добавлении файлов в архив может содержать имя, под которым файл будет записан в архиве. При остальных операциях – имя существующего файла в архиве.
НомерФайлаВАрхиве- Число целое. Имеет смысл для операций не связанных с добавлением файлов в архив.
ИмяФайлаВОС- Строка. Имя добавляемого файла или путь к папке при извлечении файлов.
МассивДанных- Двоичный массив, добавляемый в качестве файла в архив.
Комментарий- строка, комментарий, добавляемый к архиву или файлу.
Флаги- Число целое или строка, комбинация флагов:

- 1 или **KEEPPATH | СОХРАНЯТЬПУТЬ;**
- 1 или **CREATEFOLDERS | СОЗДАВАТЬПАПКИ;**
- 2 или **RECURSION | РЕКУРСИЯ;**
- 4 или **NOCASE | БЕЗРЕГИСТРА;**
- 8 или **SKIPINACCESSIBLE | ПРОПУСКАТЬНЕДОСТУПНЫЕ.**

Возвращаемое значение – Зависит от выполняемой операции. Если в процессе выполнения операции возникла ошибка, то её код и описание причины возвращается в переменных **_ERRORCODE** и **_ERRORDESCRIPTION**.

ФУНКЦИИ ДЛЯ РАБОТЫ С RAR АРХИВАМИ

Работа с архивами RAR возможна только для:

- Извлечения файлов из архива.
- Получения списка файлов и их атрибутов из архива.
- Проверка целостности архива.

RAR архив не может быть изменен или дополнен новыми файлами.

РАРОТКРЫТЬ(ИмяФайла | ДвоичныеДанные)
RAROPEN(ИмяФайла | ДвоичныеДанные)
Открывает архив для извлечения файлов или проверки целостности архива. Архив может быть файлом или двоичными данными, например, полученными по каналам связи. В случае указания имени файла будет произведена проверка наличия такого файла. Если файл существует, то он будет открыт и проверен на формат rar архива. Будет прочитан каталог файлов, что позволит получить имена, размеры и время изменения файлов в архиве. Если существующий файл не является rar архивом, то функция вернет ошибку. Двоичные данные должны представлять собой rar архив, только не записанный на носитель. В случае, если переданный массив не является правильным архивом – будет возвращена ошибка. Значение, возвращаемое при успешном открытии архива необходимо использовать в последующих вызовах функций для работы с архивом.
Предварительное открытие архива не является обязательным. Оно необходимо для получения списка и атрибутов файла. Если требуется распаковать конкретный файл, то архив можно предварительно не открывать.
После окончания работы с архивом, необходимо его закрыть функцией **РАРЗАКРЫТЬ( ) / RARCLOSE( ).**
Параметры.
ИмяФайла- Строка с именем файла архива.
ДвоичныеДанные- Двоичный массив, содержащий архив.
Возвращаемое значение – число целое. Если больше или равно нулю, то номер архива. -1 в случае возникновения ошибки. Код и описание причины ошибки возвращается в переменных **_ERRORCODE** и **_ERRORDESCRIPTION**.

РАРЗАКРЫТЬ(НомерАрхива)
RARCLOSE(НомерАрхива)
Закрывает ранее открытый архив и освобождает все занятые в связи с этим ресурсы.
Параметры.
НомерАрхива- Число целое. Значение, возвращенное функцией **РАРОТКРЫТЬ( ) / RAROPEN( ).**
Возвращаемое значение – Логическое или двоичный массив. Если архив был открыт из файла, то возвращается логическое значение. **Истина**, при успешной записи на диск, иначе, **Ложь**. Код и описание причины ошибки возвращается в переменных **_ERRORCODE** и **_ERRORDESCRIPTION**. Если архив был открыт в памяти, то возвращается двоичный массив.

РАРАРХИВ(Номер | ИмяАрхива | ДвоичныйАрхив, Операция[, ИмяФайлаВАрхиве | НомерФайлаВАрхиве [, ПутьРаспаковки|ИмяМассив[, СоздаватьКаталоги]]])
ZIP(Номер | ИмяАрхива | ДвоичныйАрхив, Операция[, ИмяФайлаВАрхиве | НомерФайлаВАрхиве [, ПутьРаспаковки|ИмяМассив[, СоздаватьКаталоги]]])
Выполняет указанную операцию с архивом. Операция может быть выполнена как с ранее открытым архивом, как часть пакета операций, так и как одиночная операция с единичным архивом. Во втором случае, функция автоматически откроет указанный архив, выполнит операцию и закроет архив с подтверждением изменений. В зависимости от параметров, функция может выполнять операции с файловыми архивами и архивами в памяти. В различных комбинациях.
Параметры.
НомерАрхива- Число целое. Значение, возвращенное функцией **РАРОТКРЫТЬ( ) / RAROPEN( ).**
ИмяАрхива- Строка. Имя файла с архивом.

ДвоичныйАрхив- Двоичный массив данных, содержащий архив. Например, данные полученные по каналам связи.

Операция- Строка, выполняемая операция. Допустимые значения:

- **FILESLIST | СПИСОКФАЙЛОВ –** Формирование массива со списком имен файлов в архиве. Имя формируемого массива передается в параметре **ИмяМассива**. Будет возвращено целое число, равное количеству отдельных записей файлов в архиве. Это могут быть как файлы, так и отдельные записи о каталогах. Если массив уже существует, то он будет предварительно очищен.
- **COUNT | КОЛИЧЕСТВОФАЙЛОВ –** Подсчет количества файлов в архиве. Будет возвращено целое число, равное количеству отдельных записей файлов в архиве. Это могут быть как файлы, так и отдельные записи о каталогах.
- **FILENAME | ИМЯФАЙЛА** – возвращает имя файла из указанного номера записи в архиве. Используются параметры: **НомерФайлаВАрхиве**. Если полученное имя будет заканчиваться косой чертой (/ или \), то это значит, что в запрошенной позиции не файл а имя каталога.
- **FILESIZE | РАЗМЕРФАЙЛА** – возвращает размер файла из указанного номера записи в архиве. Используются параметры: **НомерФайлаВАрхиве**. Для записей каталогов возвращается 0.
- **FILEINDEX | НОМЕРФАЙЛА** – возвращает номер записи файла в архиве по его имени. Используются параметры: **ИмяФайлаВАрхиве**, **Флаги**. Если параметр Флаги содержит флаг **NOCASE | БЕЗРЕГИСТРА**, то поиск производится без учета регистра букв. Если файл с указанным именем не найден в архиве, то будет возвращено значение -1.
- **EXCTRACT | ИЗВЛЕЧЬ** – извлечь файл из архива. Используются параметры: **ИмяФайлаВАрхиве | НомерФайлаВАрхиве**, **ПутьРаспаковки**, **СоздаватьКаталоги**. Если указано **ИмяФайлаВАрхиве**, не содержащее макросимволов (***** и/или **?**) или задан **НомерФайлаВАрхиве**, то функция извлечет только один указанный файл. Если параметр **СоздаватьКаталоги Истина**, то функция создаст необходимые папки относительно пути распаковки. Если параметр **ИмяФайлаВАрхиве** содержит макросимволы (***** и/или **?**), то будут извлечены все файлы, удовлетворяющие маске. При извлечении файлов могут быть созданы все необходимые каталоги. Параметр **ПутьРаспаковки** указывает путь к папке, в которую, с учетом возможного создания вложенных папок, записываются извлеченные файлы.
  - Если **ПутьРаспаковки** не указан или пустой, то результат будет возвращен в виде двоичного массива, а не записан в файл. При этом, если **ИмяФайлаыАрхиве** содержит маску, то будет извлечен только первый подходящий файл.
  - Если **ПутьРаспаковки** не пустой, то извлекаемые файлы и сопутствующая структура папок будут записаны в указанную папку.
- **TEST | ПРОВЕРИТЬ** – выполняет проверку архива. Проверка производится методом чтения всех файлов в архиве без реальной записи на диск. Возвращается значение Истина или Ложь, в зависимости от результата проверки.

ИмяФайлаВАрхиве- Строка. При добавлении файлов в архив может содержать имя, под которым файл будет записан в архиве. При остальных операциях – имя существующего файла в архиве.

НомерФайлаВАрхиве- Число целое. Имеет смысл для операций не связанных с добавлением файлов в архив.

ПутьРаспаковки- Строка. Путь к папке, в которую будут извлечены файлы.

ИмяМассива- Имя массива строк с именами файлов в архиве, который будет сформирован для операции **СПИСОКФАЙЛОВ.**

СоздаватьКаталоги- Логическое, если **Истина**, то будут создаваться все вложенные папки для извлекаемых файлов.

Возвращаемое значение – Зависит от выполняемой операции. Если в процессе выполнения операции возникла ошибка, то её код и описание причины возвращается в переменных **_ERRORCODE** и **_ERRORDESCRIPTION**.

ФУНКЦИИ ДЛЯ РАБОТЫ С ОТЛАДЧИКОМ

ОТЛАДЧИК([IPАдрес[, НомерПорта[, ИмяФайлаЖурнала]]])
DEBUGGER([IPАдрес[, НомерПорта[, ИмяФайлаЖурнала]]])
Производит подключение к серверу отладчика, зарегистрированному по указанному IP адресу и порту. Переводит вычислитель в режим отладки программного кода. Вызов функции без параметров завершает режим отладки и отключает вычислитель от сервера отладчика.
Параметры.
IPАдрес– Строка содержащая IP адрес сервера отладчика. Если не указан, то используется 127.0.0.1
НомерПорта– Число целое. Номер порта, по которому зарегистрирован сервер отладчика.
ИмяФайлаЖурнала– Строка, имя файла, в который будет производиться запись служебной информации, связанной с обменом с сервером отладчика.
Возвращаемое значение – логическое. ИСТИНА, если подключение было произведено успешно. Иначе ЛОЖЬ.

КЛЮЧЕВЫЕ СЛОВА

IF условный оператор.
Выполняет блок команд при выполнении указанного Условия или альтернативный блок команд при невыполнении Условия.
Синтаксис

```
IF (Условие)
{
       блок команд
}
[
ELSE
{
      блок команд
}
]
```

Условный оператор **IF** может быть многократно вложен один в другой, либо в другие блоки команд.

WHILE цикл по условию.
Выполняет блок команд циклически, пока выполняется указанное Условие.
Синтаксис

```
WHILE (Условие)
{
      блок команд
      [CONTINUE]
      [BREAK]
}
```

Команды внутри блока команд выполняются циклически, пока выполняется указанное Условие. Циклы по условию могут быть вложены один в другой, либо в другие блоки команд. Работа цикла может быть прервана посредством оператора **BREAK**. Для перехода к следующей итерации цикла без выполнения последующих команд используется команда **CONTINUE**.

RETURN выход из процедуры
Выполняет возврат из выполняемой процедуры или функции.
Синтаксис

```
RETURN [(Возвращаемое значение)];
```

При выходе из функции Возвращаемое значение будет являться значением этой функции.

BREAK выход из последнего цикла WHILE
Прерывает выполнение цикла **WHILE** и передает управление оператору, следующему за блоком команд цикла **WHILE.**
Синтаксис
BREAK;

TRY...CATCH обработка исключений.
Выполняет блок команд. Если во время выполнения блока команд возникло исключение, то обработка передается блоку команд CATCH. Исключениями являются любые ошибки, кроме синтаксических, возникающие на этапе трансляции исходного текста, приводящие к прекращению выполнения блока команд TRY.
Синтаксис

```
TRY
{
      блок команд
}
CATCH
{
      блок команд
}
```

CONTINUE Передача управления следующей итерации цикла WHILE.
Начинает новую итерацию цикла **WHILE** без выполнения последующих команд блока команд цикла.
Синтаксис
CONTINUE;

PUBLIC объявление глобальных переменных.
Объявляет указанные переменные глобальными.
Синтаксис
PUBLIC ИмяПеременной1[, ИмяПеременной2[,...]]
Указанные в списке переменные объявляются глобальными. Если на момент обработки команды указанные переменные уже существуют, то они переносятся в список глобальных переменных, иначе, создаются новые переменные с указанными именами и пустыми значениями.

FUNCTION объявление функции пользователя.
ФУНКЦИЯ объявление функции пользователя.
Объявляет функцию пользователя.
Синтаксис

```
FUNCTION [LOCAL | ЛОКАЛЬНАЯ] ИмяФункции([ИмяПараметра1[, ИмяПараметра2[,...]]])
{
      блок команд;
}
```

Указанная в описании функция добавляется в список пользовательских функций. Если в этом списке уже есть функция с указанным именем, то она будет заменена на новую. Встроенные функции не могут быть переопределены указанным способом. Если имени функции предшествует слово **LOCAL** или **ЛОКАЛЬНАЯ**, то при вызове такой функции будут доступны все внешние локальные переменные. В противном случае, такие переменные не будут доступны.

ТОЧКАОСТАНОВКИ
BREAKPOINT
Прерывание выполнения программы и передача управления отладчику.
Синтаксис
ТОЧКАОСТАНОВКИ;

В случае, если вычислитель не подключен к отладчику, то команда игнорируется.

ПОНЯТИЕ
CONCEPT
Добавляет указанное понятие в систему.
Синтаксис
CONCEPT @ИмяПонятия= { Описание понятия в формате XML }
XML описание может содержать следующие тэги:
<Метка> - метка понятия.
<ИмяПоля> - имя поля. Тэг указывает, что понятие является базовым.
<Выражение> - текст выражения. Тэг указывает, что понятие является Выражением.
<Запрос> - SQL запрос. Тэг указывает, что понятие является SQL выражением.
<Тип> - тип значения понятия.
<Результат> - имя поля результата SQL запроса.
Пример.
CONCEPT @БанкГрузополучателя= {<Метка>Банк грузополучателя</Метка><Запрос>SELECT name FROM sprbank WHERE code= (SELECT bank FROM sprclient WHERE code= @Грузополучатель) </Запрос><Результат>name</Результат> <Тип>Символьный(240)</Тип>}

КОНСТАНТА
CONSTANT
Добавляет указанную константу в систему.
Синтаксис
CONSTANT _ИмяКонстанты= Выражение
Команда добавляет в систему константу с указанным именем и значением.

SQL[(Имяконтекста][,СписокКодовФилилов|НомерСоединения[,ИмяВременнойТаблицы [,ПризнакОткатаТранзакции]]])]
Выполнение блока запросов SQL с последующим объединением данных и обработкой ошибок.
Синтаксис
SQL[(Имяконтекста][,СписокКодовФилилов|НомерСоединения[,ИмяВременнойТаблицы[, ПризнакОткатаТранзакции]]])]
{ SQL запросы }
SQLAGGREGATE
{ запрос формирующий итоговый результат }
SQLERROR
{ команды обработки ошибок }
Параметры.

ИмяКонтекста	- Строка, если указана, то результат выполнения запросов будет помещен в указанный контекст.
СписокКодовФилиалов	- Строка, состоящая из списка кодов, разделенных запятыми или пробелами, в базах данных которых необходимо выполнить SQL запросы. Если вместо кодов филиалов будет указан символ *, то список будет составлен автоматически, и будет содержать коды филиалов, у которых указано имя базы данных или адрес, для прямого удаленного подключения.
НомерСоединения	- Число целое, номер соединения, в базе данных которого необходимо выполнить SQL запросы.
ИмяВременнойТаблицы	- Строка, имя временной таблицы, в которую будут собираться результаты выполнения запросов для последующего выполнения агрегации данных.
ПризнакОткатаТранзакции	- Логическое, если истина, то в случае возникновения ошибок при выполнении запросов, SQL серверу будет подана команда **ROLLBACK.**

Команда выполняет указанный блок запросов в рабочей базе данных, или в базах данных перечисленных филиалов. В случае указания номера соединения, запросы будут выполнены в базе данных, связанной с соединением. Если указан агрегирующий

запрос, то должно быть указано имя временной таблицы, в которую будут добавляться записи, являющиеся результатом выполнения основных запросов. Имена и количество полей временной таблицы должны совпадать с полями результата выполнения запросов. Если указан агрегирующий запрос, то он будет выполнен в рабочей базе данных, после того, как будут собраны данные со всех филиалов. Результат выполнения агрегирующего запроса будет являться результатом выполнения всей команды. Если указано имя контекста, то из результата выполнения команды будет создан контекст. Если имя контекста не указано, то результатом выполнения команды будет являться значение первого поля первой строки итоговой выборки. Если выполнение запросов привело к ошибкам, то будет выполнен блок команд из части SQLERROR.
Части команды, содержащие SQL запросы могут быть параметризованы вычисляемыми выражениями, значения которых будут помещены в указанные места запросов. Выражения необходимо вводить ограничив их двойными квадратными скобками. В нижеследующем примере переменная **ВременнаяТаблица** использована как параметр запроса в части **SQLAGGREGATE**.

Пример. В случае успешного выполнения следующего блока команд в переменную МаксимальныйИД будет записано максимальное значение поля identity_column из всех таблиц spectree, находящихся во всех доступных филиалах. Базы данных филиалов могут располагаться как на локальном SQL сервере, так и на другом SQL сервере, или быть связаны через удаленное соединение службы Айтида.

```
ВременнаяТаблица      = "##" + УникальноеИмя( );
ЗАПРОС( "CREATE TABLE " + ВременнаяТаблица + " ( ident int )" );

МаксимальныйИД        =
SQL ( "", "*", ВременнаяТаблица )
{
       SELECT identity_column AS ident FROM spectree
}
SQLAGGREGATE
{
       SELECT MAX( ident ) FROM [[ВременнаяТаблица]]
}
SQLERROR
{
       Сообщение( "Ошибка выполнения запроса" );
       RETURN -1;
};
```