

Строковые функции

Chr

Chr(Аргумент)

- Аргумент — числовой код символа Unicode в диапазоне [0, 65535].

Возвращает символ Unicode по его коду.

Concat

Concat(Строка, Строка [, ... Строка])

- Строка — строковое выражение.

Объединяет несколько строк в одну. Пропущенные значения интерпретируются как строки с нулевой длиной, т.е. как "".

Count

Count(Строка)

- Строка — строковое выражение.

Возвращает количество символов в строке.

DamLevDist

DamLevDist(Строка1, Строка2)

- Строка1, Строка2 — выражения строкового типа.

Функция возвращает значение расстояния Дамерау-Левенштейна для строк Строка1, Строка2. Расстояние Дамерау-Левенштейна также называют расстоянием редактирования с учетом перестановок, которое является мерой похожести двух строк. Результат — это минимальное количество операций удалений, вставки, замены и перестановок символов, которое нужно произвести, чтобы преобразовать одну строку в другую.

Find

Find(Подстрока, Строка[, Порядок = 1])

- Подстрока — искомая подстрока,
- Строка — строка, в которой ищется подстрока,
- Порядок — необязательный параметр, позицию какого по счету вхождения Подстроки в Строку возвращать. Аргумент числового типа.

Функция возвращает позицию первого символа Подстроки заданного Порядка в Строке. Порядок может быть как положительным, так и отрицательным. Если Порядок положительный, то поиск происходит слева направо, если отрицательный, то справа налево. Если вхождения заданного порядка нет, то функция возвращает 0. Например, Find("A", "A000A00A", 2) вернет позицию второго слева вхождения подстроки "A" в строку "A000A00A", т.е. 5; Find("A", "A000A00A", -1) вернет позицию первого справа вхождения подстроки "A" в строку "A000A00A", т.е. 8. Если Порядок равен нулю, то функция возвращает Неопределенное значение (Null).

Для аргумента Подстрока недопустимо значение `null` .

Format

Format(Строка_формата, Выражение [, ... Выражение])

- Строка_формата — строковое выражение специального вида
- Выражение — произвольное строковое выражение.

Возвращает список выражений в заданном формате. Строка_формата имеет следующий вид: [Строка] %s [Строка] [... [[Строка]%s[Строка]]] Строка — строковое выражение %s — символ, на место которого ставится очередное Выражение из списка.

Left

Left(Строка, Количество_символов)

- Строка — выражение строкового типа.

Возвращает заданное количество символов левой части строки.

Для аргумента Количество_символов недопустимо значение `null` .

LevDist

LevDist(Строка1, Строка2)

- Строка1, Строка2 — выражения строкового типа.

Функция возвращает значение расстояния Левенштейна для строк Строка1, Строка2. Расстояние Левенштейна также называют расстоянием редактирования, которое является мерой схожести двух строк. Результат — это минимальное количество операций удалений, вставки и замены символа, которое нужно произвести, чтобы преобразовать одну строку в другую.

Lower

Lower(Аргумент)

- Аргумент — строковое выражение.

Возвращает аргумент в нижнем регистре.

RegexDomain

RegexDomain(Вх.Строка [, Ном.Вхожд = 1])

- Вх.Строка — входная строка, в которой будет искаться Email-адрес;
- Ном.Вхожд — порядковый номер Email-адрес-а, в искомой строке.

Возвращает Ном.Вхожд-й адрес сайта из строки Вх.Строка не включая www до домена 1-го уровня включительно. Для поиска используются регулярные выражения.

Для аргумента Ном.Вхожд недопустимо значение `null`.

RegexEmail

RegexEmail(Вх.Строка [, Ном.Вхожд = 1])

- Вх.Строка — входная строка, в которой будет искаться Email-адрес;
- Ном.Вхожд — порядковый номер Email-адрес-а, в искомой строке.

Извлекает Ном.Вхожд-й Email-адрес из строки Вх.Строка. Для поиска используются регулярные выражения.

Для аргумента Ном.Вхожд недопустимо значение `null`.

RegexMatch

RegexMatch(Рег.Выр., Строка)

- Рег.Выр. — строка регулярного выражения;
- Строка — входная строка.

Проверяет соответствие строки регулярному выражению. Используется библиотека PCRE2. Синтаксис и семантика совместимы с регулярными выражениями Perl.

RegexMatchCount

RegexMatchCount(Рег.Выр., Строка)

- Рег.Выр. — строка регулярного выражения;
- Строка — входная строка.

Возвращает кол-во частей строки, соответствующих регулярному выражению. Используется библиотека PCRE2. Синтаксис и семантика совместимы с регулярными выражениями Perl.

RegexMatchedExp

RegexMatchedExp(Рег.Выр., Строка [, НомерЧасти = 1])

- Рег.Выр. — строка регулярного выражения;
- Строка — входная строка;
- НомерЧасти — порядковый номер части, соответствующей регулярному выражению, входной строки. Параметр должен входить в диапазон от 1 до RegexMatchCount.

Возвращает часть строки, соответствующую регулярному выражению. Используется библиотека PCRE2. Синтаксис и семантика совместимы с регулярными выражениями Perl.

Для аргумента НомерЧасти недопустимо значение `null` .

RegexMatchedNamedSubExp

RegexMatchedNamedSubExp(Рег.Выр., Строка, ИмяГруппы [, НомерЧасти = 1])

- Рег.Выр. — строка регулярного выражения;
- Строка — входная строка;
- ИмяГруппы — имя группы, определяемой метасимволами (...);
- НомерЧасти — порядковый номер части, соответствующей регулярному выражению, входной строки. Параметр должен входить в диапазон от 1 до RegexMatchCount.

Возвращает группу части строки, соответствующую регулярному выражению. Группы нумеруются слева направо, в порядке появления открывающихся скобок. Используется библиотека PCRE2. Синтаксис и семантика совместимы с регулярными выражениями Perl.

Для аргумента НомерЧасти недопустимо значение `null` .

RegexMatchedSubExp

RegexMatchedSubExp(Рег.Выр., Строка, НомерГруппы [, НомерЧасти = 1])

- Рег.Выр. — строка регулярного выражения;
- Строка — входная строка;

- **НомерГруппы** — порядковый номер группы, определяемой метасимволами (...);
- **НомерЧасти** — порядковый номер части, соответствующей регулярному выражению, входной строки. Параметр должен входить в диапазон от 1 до `RegexMatchCount`.

Возвращает группу части строки, соответствующую регулярному выражению. Группы нумеруются слева направо, в порядке появления открывающихся скобок. Используется библиотека PCRE2.

Синтаксис и семантика совместимы с регулярными выражениями Perl.

Для аргументов **НомерГруппы** и **НомерЧасти** недопустимо значение `null` .

RegexReplace

RegexReplace(Рег.Выр., Вх.Строка, СтрокаЗамены [, НомерЧасти = 1, Расш.Синтаксис = False])

- **Рег.Выр.** — строка регулярного выражения;
- **Вх.Строка** — входная строка, которая будет подвержена замене;
- **СтрокаЗамены** — строка, которая будет вставлена вместо части входной строки, соответствующей регулярному выражению;
- **НомерЧасти** — порядковый номер части, соответствующей регулярному выражению, входной строки. Параметр должен входить в диапазон от 1 до `RegexMatchCount`.
- **Расш.Синтаксис** — расширенный синтаксис замены (только для PCRE2)

Возвращает **Вх.Строка**, в которой **НомерЧасти**-ое вхождение выражения **Рег.Выр.** будет заменено на **СтрокаЗамены**. Используется библиотека PCRE2. Синтаксис и семантика совместимы с регулярными выражениями Perl.

Для аргументов **НомерЧасти** и **Расш.Синтаксис** недопустимо значение `null` .

RegexReplaceAll

RegexReplaceAll(Рег.Выр., Вх.Строка, СтрокаЗамены [, Расш.Синтаксис = False])

- **Рег.Выр.** — строка регулярного выражения;
- **Вх.Строка** — входная строка, которая будет подвержена замене;
- **СтрокаЗамены** — строка, которая будет вставлена вместо частей входной строки, соответствующих регулярному выражению.
- **Расш.Синтаксис** — расширенный синтаксис замены (только для PCRE2)

Возвращает **Вх.Строка**, в которой все вхождения выражения **Рег.Выр.** будут заменены на **СтрокаЗамены**. Используется библиотека PCRE2. Синтаксис и семантика совместимы с регулярными выражениями Perl.

Для аргумента **Расш.Синтаксис** недопустимо значение `null` .

Repeat

Repeat(Строка, Количество_повторов)

- Строка — строковое выражение.
- Количество_повторов — целое выражение

Возвращает строку — повторяющую заданную строку необходимое количество раз.

Для аргумента Количество_повторов недопустимо значение `null` .

Replace

Replace(Строка, Старый_шаблон, Новый_шаблон [, Заменять_все, Игнорировать_регистр])

- Строка — строка, в которой необходимо сделать замену;
- Старый_шаблон — заменяемая подстрока;
- Новый_шаблон — строка, на которую будет заменяться Старый шаблон;
- Заменять_все — флаг замены всех вхождений Старого шаблона на Новый шаблон. Аргумент логического типа;
- Игнорировать_регистр — флаг чувствительности к регистру при замене. Аргумент логического типа.

Функция возвращает Строку, в которой вхождения Старого шаблона были заменены на Новый шаблон. Если флаг "Заменять все" имеет значение True (истина), то в возвращаемой строке будут заменены все вхождения Старого Шаблона на Новый шаблон. Если флаг "Заменять все" имеет значение False (ложь), то в возвращаемой строке будет заменено только первое по порядку вхождение. Если флаг "Игнорировать регистр" имеет значение True (истина), то поиск заменяемых фрагментов (Старый шаблон) будет регистронезависимым. Если флаг "Игнорировать регистр" имеет значение False (ложь), то поиск заменяемых фрагментов (Старый шаблон) будет регистрозависимым. Последние два параметра можно не указывать. Если не указан флаг "Игнорировать регистр", то поиск заменяемых фрагментов будет регистрозависимым . Если не указан флаг "Заменять все", то будет заменено только первое вхождение заменяемой подстроки. Например, `Replace("A000a00A", "A", "B", True, True)` вернет строку `"B000B00B"` , т.е. заменит все вхождения строки "A" на строку "B" игнорируя регистр символов; `Replace("A000a00A", "A", "B", True, False)` вернет строку `"B000a00B"`. `Replace("A000a00A", "A", "B")` вернет строку `"B000a00A"`.

Для аргументов Старый_шаблон, Новый_шаблон, Заменять_все и Игнорировать_регистр недопустимо значение `null` .

Right

Right(Строка, Количество_символов)

- Строка — выражение строкового типа.

Возвращает заданное количество символов правой части строки.

Для аргумента Количество_символов недопустимо значение `null` .

Space

Space(Количество_пробелов)

- Количество_пробелов — целое выражение.

Возвращает заданное количество пробелов.

Str

Str(Аргумент)

- Аргумент — выражение числового типа.

Возвращает строковое представление Аргумента с учетом локали.

StrZero

StrZero(Выражение, Общая_длина)

- Выражение — числовое выражение.
- Общая_длина — длина результирующей строки

Добавляет спереди заданного выражения нули так, чтобы длина результирующей строки равнялась Общей_длине.

Stuff

Stuff(Источник, Начало_позиции, Длина [, Вставляемая_строка = ""])

- Источник — строковое выражение.
- Вставляемая_строка — строковое выражение.
- Начало — позиция начала вставки
- Длина — длина удаляемой подстроки

Возвращает строку следующего вида. Сначала из Источника удаляется подстрока, начиная с позиции Начало длиной Длина. Затем на место удаленной подстроки вставляется Вставляемая_строка.

Для аргументов Начало_позиции, Длина и Вставляемая_строка недопустимо значение `null` .

SubStr

SubStr(Источник, Начало, Длина)

- Источник — строковое выражение.
- Начало — позиция начало подстроки
- Длина — длина подстроки

Возвращает подстроку из строки — источника, начиная с позиции Начало длиной Длина.

Для аргументов Начало и Длина недопустимо значение `null` .

Trim

Trim(Аргумент)

- Аргумент — строковое выражение.

Возвращает Аргумент без ведущих и ведомых пробелов (удаляются символы с кодом от 0 до 32 включительно). Если Аргумент содержит только пробелы — возвращает пустую строку.

TrimLeft

TrimLeft(Аргумент)

- Аргумент — строковое выражение.

Возвращает Аргумент без ведущих пробелов (удаляются символы с кодом от 0 до 32 включительно). Если Аргумент содержит только пробелы — возвращает пустую строку.

TrimRight

TrimRight(Аргумент)

- Аргумент — строковое выражение.

Возвращает Аргумент без ведомых пробелов (удаляются символы с кодом от 0 до 32 включительно). Если Аргумент содержит только пробелы — возвращает пустую строку.

Upper

Upper(Аргумент)

- Аргумент — строковое выражение.

Возвращает аргумент в верхнем регистре.

Val

Val(Строка)

- Строка — строковое выражение.

Преобразует заданную строку в число с учетом локали