**Язык М Power Query. Типы – Введение и текст (строки)**

Изучите запрос Power Query под микроскопом. Что вы увидите? Данные, пульсирующие между выражениями. Увеличьте масштаб. Вы различите отдельные элементы данных, составляющие этот поток. Элементы данных группируются в зависимости от типа содержащихся в них значений: некоторые содержат текст, другие – дату и время, третьи – значения ИСТИНА или ЛОЖЬ. Конечно же, куда без чисел!?

В языке M поддерживается довольно много типов значений. Типам свойственно особое поведение. Например, для даты и времени существует специальное правило сложения: добавьте время к дате, и вы получите дату-время. Описание всех типов занимает много места. В этой заметке мы рассмотрим специфику текста (строк). Другие типы оставим на потом.[[1]](#footnote-1)

[Предыдущая заметка](https://baguzin.ru/wp/?p=25095) Следующая заметка

#### Литералы

Отступим на шаг назад и упомянем, что основными синтаксическими элементами языка М являются идентификаторы, ключевые слова, литералы, операторы, знаки препинания, комментарии и пробелы. Некоторые элементы мы обсудили ранее. *Литералом* в коде М (как и в других языках программирования) называется фиксированное значение (подробнее см. определение в [Википедии](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%82%D0%B5%D1%80%D0%B0%D0%BB_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) и в [спецификации](https://docs.microsoft.com/ru-ru/powerquery-m/m-spec-lexical-structure) языка М).

"Привет!" // пример текстового литерала

100.3 // пример числового литерала (в коде М десятичный разделитель – точка)

Разные типы имеют разные правила литерального синтаксиса.

#### Текстовые литералы

В M-коде для ввода текста у вас есть только одна возможность – текст [Юникода](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4#:~:text=%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%CC%81%D0%B4%20(%D1%87%D0%B0%D1%89%D0%B5%20%D0%B2%D1%81%D0%B5%D0%B3%D0%BE)%20%D0%B8%D0%BB%D0%B8%20%D0%A3%D0%BD%D0%B8%D0%BA%D0%BE%CC%81%D0%B4,%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%20%D1%8F%D0%B2%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D0%BB%D0%B0%D0%B4%D0%B0%D1%8E%D1%89%D0%B8%D0%BC%20%D0%B2%20%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%D0%B5.) в двойных кавычках.

"Hello, World!"

Если текст должен содержать кавычки, экранируйте кавычки, удваивая их:

"He said ""run"" and so I did" // возвращает строку *He said "run" and so I did*

Текст может занимать несколько строк.

"Hello

World"

Если вам нужно разместить текст в одной строке кода, а результат отразить в нескольких строках, используйте escape-последовательности. Они включают escape-символ, окруженный круглыми скобками и знаком # в начале. Power Query поддерживает три escape-символа: cr – возврат каретки, lf – перевод строки, tab – табуляция. Например…

**Листинг 1[[2]](#footnote-2)**

= "Hello#(lf)World""



Рис. 1. escape-символ перевода строки разбивает текст на две строки

Можно объединить несколько escape-кодов внутри escape-последовательности, разделив их запятыми. Никакие дополнительные пробелы не допускаются. Любопытно, что это одно из немногих мест в Power Query, где пробелы между элементами имеют значение. Как правило, можно использовать любое число пробелов для повышения удобочитаемости кода.

**Листинг 2**

= "Hello#(cr,lf)World"

Редактор PQ автоматически изменил эту строку на…

= "Hello#(cr)#(lf)World"

Такое впечатление, что имеется какая-то недокументированная особенность, и возврат каретки (cr) вообще не оказывает влияние на результат:



Рис. 2. Работает ли возврат каретки в М?

Более того, использование #(cr,lf) может приводить к некорректному отображению после загрузки запроса на лист Excel. Рекомендую не использовать #(cr), ни в одиночку, ни в паре с другими escape-символами.

Escape-последовательность также можно использовать для ввода шестнадцатеричных значений Юникода. Это может пригодиться для символов недоступных с клавиатуры. Например…

**Листинг 4**

= "Участник #(246B)"



Рис. 3. Вывод символа недоступного с клавиатуры

Если, наконец, вы хотите, чтобы сама escape-последовательность отразилась в строке, используйте escape-код, называемый escape escape – символ # в скобках сразу за символом #:

**Листинг 5**

= "Код детали #(#)(2501)"



Рис. 4. Вывод escape-последовательности, как элемента текстового литерала

#### Операторы

Со значениями типа text вы можете работать, используя несколько операторов. При этом не важно получили ли вы переменную с типом текст вручную с помощью строкового литерала или в качестве результата другого выражения. Тип text поддерживает работу со стандартными операторами сравнения (=, <>, >=, >, <=, <).

"a" = "a" // возвращает значение true

"a" > "b" // возвращает значение false

Сравнение чувствительно к регистру. Чтобы сравнить без учета регистра, вам нужно использовать библиотечную функцию [Comparer.Equals](https://docs.microsoft.com/ru-ru/powerquery-m/comparer-equals).

**Листинг 6**

Comparer.Equals(Comparer.OrdinalIgnoreCase, "A", "a") // возвращает значение true

#### Сцепление

Текстовые значения могут быть объединены с помощью оператора конкатенации &:

= "Good" & " " & "Morning!" // возвращает *Good Morning!*

Однако Power Query при объединении нескольких литералов автоматически не преобразует нетекстовые значения в текст. В частности, литералы с типом text и number не могут быть объединены. Выражение…

**Листинг 7**

= "You have " & 5 & " left"

…вернет ошибку:



Рис. 5. Литералы с типом text и number нельзя объединить

Чтобы выполнить объединение, воспользуйтесь библиотечной функцией для преобразования числа в текст…

**Листинг 8**

= "You have " & Text.From(0.5) & " left"

… или …

**Листинг 9**

= "You have " & Number.ToText(0.5, "P") & " left"



Рис. 6. Число, преобразованное в текст

Функция [Number.ToText](https://docs.microsoft.com/ru-ru/powerquery-m/number-totext) более гибкая. Она позволяет во втором аргументе задать формат преобразования. Мы, в частности, использовали параметр Р, который указывает, что следует вывести проценты, т.е., число, умноженное на 100 и отображаемое со знаком процента.

Конкатенация текста и значения null возвращает null.

= "Something Profound" & null // возвращает null

= null & "Something Profound" // возвращает null

Более гибко ведет себя функция [Text.Combine](https://docs.microsoft.com/ru-ru/powerquery-m/text-combine), которая игнорирует значения null. Код…

**Листинг 12**

= Text.Combine({"Something Profound", null})

…вернет…



Рис. 7. Функция Text.Combine более робастна, чем конкатенация, так как игнорирует значения null

Библиотека Power Query включает в себя довольно много [функций](https://docs.microsoft.com/ru-ru/powerquery-m/text-functions) для работы с текстом. К сожалению на данный момент поддержка [регулярных выражений](https://baguzin.ru/wp/?p=24122) не входит в этот список.

1. Заметка написана на основе статьи [Ben Gribaudo. Power Query M Primer (Part 6): Types—Intro & Text (Strings)](https://bengribaudo.com/blog/2018/06/26/4470/power-query-m-primer-part6-types-intro-text). Если вы впервые сталкиваетесь с Power Query, рекомендую начать с [Марк Мур. Power Query](http://baguzin.ru/wp/?p=19403). [↑](#footnote-ref-1)
2. Номер листинга соответствует номеру запроса в приложенном Excel файле. [↑](#footnote-ref-2)