**Язык М Power Query. Переменные и идентификаторы**

В предыдущих заметках мы использовали переменные, не вдаваясь в их специфику. Такой подход был основан на предположении, что у вас есть как минимум небольшой опыт работы с другими языками программирования. Однако, переменные в языке M Power Query ведут себя весьма специфически. В настоящей заметке мы глубже изучим концепции, связанные с переменными, постараемся прояснить сделанные ранее неявные допущения.[[1]](#footnote-1)

[Предыдущая заметка](https://baguzin.ru/wp/?p=25008) Следующая заметка

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1. Допустимые имена обычных идентификаторов

### Что мы уже знаем

[В первой части](https://baguzin.ru/wp/?p=24951) мы говорили о выражениях *let*, которые позволяют вычислять значение с использованием одного или нескольких промежуточных выражений. Вместо того, чтобы писать одно большое длинное выражение, *let* позволяет разделить код на несколько шагов. Каждый шаг возвращает значение, а в итоге формируется значение, возвращаемое всем выражением *let*. Переменные используются для каждого шага, а также для [функций](https://baguzin.ru/wp/?p=24960), которые вы, возможно, захотите определить внутри запроса. Например…

**Листинг 1[[2]](#footnote-2)**

let

SalesTotal = 100 + 15 + 275 + 25,

CommissionRate = 0.2,

CalculateCommission = (sales, rate) => sales \* rate,

Commission = CalculateCommission(SalesTotal, CommissionRate),

Result = Commission

in

Result

У каждой переменной есть имя (например, *CommissionRate*). Давайте начнем с имен…

### Идентификаторы

#### Имена идентификаторов

Просматривая код M, созданный редактором Power Query, вы видели, что синтаксис имен переменных необычен. Когда вы сами пишете код, то используете что-то типа *variableName*, а в сценариях, созданных графическим интерфейсом PQ, переменные начинаются со знака решетки # и взяты в кавычки:

#"Добавлен пользовательский объект"

Почему так?

Мы кодируем имена переменных как обычные идентификаторы. Графический интерфейс PQ создает так называемые идентификаторы в кавычках.

#### Типы идентификаторов

##### Обычный идентификатор

Обычный идентификатор должен начинаться, либо с буквы, либо с символа подчеркивания. Обычные идентификаторы не должны содержать пробелов, ключевых слов M и некоторых других специальных символов (см. рис. 1).

##### Идентификаторы в кавычках

В идентификаторе, заключенном в кавычки, допускается использование любых символов: пробелов, ключевых слов M, знаков комментария. Внутри идентификаторов в кавычках можно использовать даже другие кавычки. Их нужно экранировать, удваивая кавычки.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2. Допустимые имена идентификаторов в кавычках

В идентификаторах в кавычках можно даже использовать пустую строку в качестве идентификатора!

**Листинг 2**

let

#"" = 1

in

#""

В большинстве языков программирования пробелы в именах переменных не используются. В M пробелы полезны. Дело в том, что имена на панели ПРИМЕНЕННЫЕ ШАГИ являются именами переменных из выражения *let*. Пробелы в этих именах облегчают чтение.

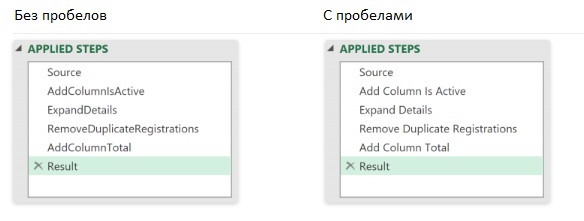


Рис. 3. Пробелы в именах переменных облегчают чтение шагов в интерфейсе редактора Power Query

##### Обобщенные идентификаторы

Третий тип идентификатора, обобщенный идентификатор (generalized identifier), представляет собой нечто среднее между обычными идентификаторами и идентификаторами в кавычках. Этот тип идентификатора допускается только в квадратных скобках, например, при ссылке на поле записи или имя столбца. Поскольку квадратные скобки позволяют определить, где начинается и заканчивается идентификатор, разрешены определенные символы и ключевые слова, которые в противном случае могут использоваться только в идентификаторах, заключенных в кавычки. Например, обобщенные идентификаторы допускают пробелы между словами, хотя пробелы перед первым словом или после последнего слова игнорируются.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 4. Допустимые имена обобщенных идентификаторов

Игнорирование пробелов в начале или конце имени обобщенного идентификатора может представлять проблему, если эти пробелы есть в заголовках столбцов исходных данных. Тех данных, которые импортируются в Power Query. Представьте, что в исходных данных есть столбец с заголовком "First Name " (с пробелом в конце). Следующий код вернет ошибку:

**Листинг 3**

let

Источник = #table({"First Name "}, {{"Joe"}})

in

Источник[First Name ]

Изображение выглядит как текст

Автоматически созданное описание

Рис. 5. Игнорирование пробелов в конце заголовка столбца приводит к ошибке, если использовать обобщенные идентификаторы

Что произошло? Идентификатор [First Name ] проигнорировал пробел в конце имени, и попытался вывести значения столбца [First Name] (без пробела в конце). Не нашел такого имени столбца, и вернул ошибку.

Надежный путь – использовать идентификаторы в кавычках:

**Листинг 4**

let

Source = #table({"First Name "}, {{"Joe"}})

in

Source[#"First Name "]

Изображение выглядит как текст

Автоматически созданное описание

Рис. 6. Идентификаторы в кавычках работают всегда

#### Идентификация различных сущностей, а не только переменных

Узнав об обобщенных идентификаторах, вы можете возразить: «Мы говорили об идентификации переменных. Имена полей и столбцов не являются переменными!» Оказывается, что идентификаторы применяются не только для переменных. То, что они идентифицируют, определяется контекстом, в котором они используются. До сих пор мы использовали их для идентификации переменных, однако, когда они используются в контексте квадратных скобок, они идентифицируют поля записей или столбцы таблицы.

#### Идентификатор – ссылка

Идентификатор – это не однозначно заданное имя, а лишь ссылка на поименованный объект. Когда один и тот же идентификатор используется в различных синтаксических вариантах, все варианты ссылаются на одно и то же. Например, допустим такой код…

**Листинг 5**

let

Weight = 50

in

#"Weight"

#### Область определения переменных

Если в выражении определено подвыражение, то переменные в подвыражении могут ссылаться на все переменные выражения. Наоборот не работает: переменные выражения не могут ссылаться на переменные, определенные внутри подвыражения. Следующий код вернет значение 60.

**Листинг 6**

let

a = 10, // эта переменная "видит" b и c (может на них ссылаться), но не видит d

b =

let

d = a, // эта переменная "видит" a и c

Result = d + 25

in

Result,

c = 15 // эта переменная "видит" a и b

in

a + b + c

Если же вы измените последнюю строку на…

a + b + c + d

… получите ошибку:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 7. Внешнее выражение не видит переменные в подвыражении

Из этого свойства следует интересное следствие. Переменные в подвыражении могут иметь те же имена, что и во внешнем выражении. Если внутри подвыражения переменные не определены, PQ попытается найти определение переменной во внешнем выражении. Листинг 7 вернет значение 60.

**Листинг 7**

let

a = 10,

b =

let

Result = a + 25

in

Result,

c = 15

in

a + b + c

Если переменная определена в подвыражении, PQ будет считать, что эти идентификаторы ссылаются на разные объекты. Листинг 8 вернет значение 55.

**Листинг 8**

let

a = 10,

b =

let

a = 5,

Result = a + 25

in

Result,

c = 15

in

a + b + c

#### Область определения для обобщенных идентификаторов

Посмотрите выше, что мы говорили об обобщенных идентификаторах в квадратных скобках. Рассматривайте область определения переменных внутри скобок, как вложенную по отношению к области за пределами скобок:

[

A = 1, // может ссылаться на B (включая B[BB]) и C

B = [ BB = A ], // может ссылаться на А и С

C = 2 // может ссылаться на А и В (включая B[BB])

]

Здесь B – имя таблицы, а ВВ – имя столбца таблицы В.

#### Рекурсия

Как мы говорили [ранее](https://baguzin.ru/wp/?p=25008), для ссылки выражения на идентификатор, которому присвоено само выражение, добавьте @ перед ссылкой. Ссылаться в выражении на само себя удобно при определении рекурсивных функций.

**Листинг 9**

let

SumConsecutive = (x) => if x <= 0 then 0 else x + @SumConsecutive(x - 1),

Result = SumConsecutive(4)

in

Result

Здесь выражение…

if x <= 0 then 0 else x + @SumConsecutive(x - 1)

… присвоено идентификатору SumConsecutive.

### В следующей заметке

Вы изучили, как определять переменные. Обратили ли вы внимание, что в коде М нельзя изменить значение переменной после ее первоначального определения? Следующий код вернет столь сильную синтаксическую ошибку, что в Расширенном редакторе даже нельзя нажать *Готово* (я сохранил код поставив перед строкой *с = 15* знак комментария):

**Листинг 10**

let

a = 10,

b =

let

Result = a + 25

in

Result,

c = 15,

c = c + 5

in

a + b + c

Изображение выглядит как текст

Автоматически созданное описание

Рис. 8. Фатальная ошибка при попытке дать ранее определенной переменной новое значение

В самом коде М нельзя изменить значение ранее определенной переменной, но… Если выражение, присвоенное переменной, запрашивает внешние данные, и при обновлении каждый раз возвращает новое значений, проблем нет.

Невозможность изменять значение переменной внутри кода М, приводит к интересному следствию. В коде M нет классических циклов. Например, привычного для VBA *For… Next*. Поскольку код внутри цикла изменяет переменную (счетчик), в M такое не работает.

Невозможность переопределения переменных в коде М является важной особенностью языка. В следующей заметке мы обсудим связанные с этим понятия *последовательность оценки* и *ленивая оценка*.

1. Заметка написана на основе статьи [Ben Gribaudo. Power Query M Primer (Part 4): Variables & Identifiers](https://bengribaudo.com/blog/2018/01/19/4321/power-query-m-primer-part4-variables-identifiers). 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)