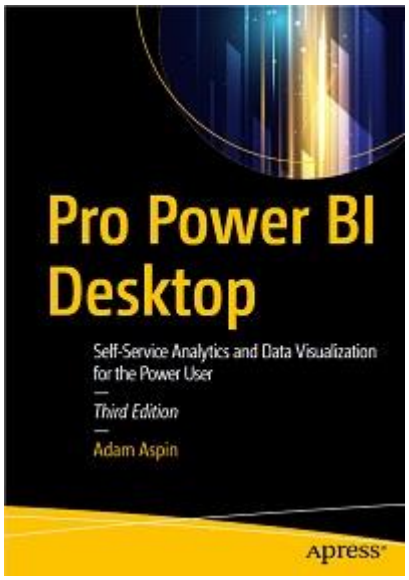
Адам Аспин. Язык M Power Query

Это сокращенный перевод главы из книги Адама Аспина, посвященной работе в Power BI. M – это язык запросов, используемый Power BI и Excel Power Query. Описаны многообразные функции языка M, типы данных, создание пользовательской функции, добавление комментариев и многое другое. Заметка будет полезна, как быстрое введение в язык M Power Query.

Adam Aspin. Pro Power BI Desktop: Self-Service Analytics and Data Visualization for the Power User. – New York: Apress Media, 2020. – 918 p.



Большинству пользователей – и большую часть времени – вряд ли вообще нужно будет использовать язык M напрямую. Для многих задач будет вполне достаточно интерфейса редактора Power Query. Тем не менее, иногда может понадобиться:

- добавить функции, которые не доступны через графический интерфейс,
- добавить логику в код или сгенерировать последовательности дат/чисел,
- создать собственные списки, записи или таблицы или управлять ими программным путем,
- создать пользовательскую функцию,
- использовать расширенный редактор для изменения кода,
- добавить комментарии.

Прежде чем начать, несколько предостережений:

- На первый взгляд Язык M может показаться заумным.
- Документация чрезвычайно технична и не очень понятна непосвященным.
- Кривая обучения может быть крутой даже для опытных программистов.
- Язык M сильно отличается от VBA, который хорошо знают многие опытные пользователи Excel.
- Настройка шага вручную может привести к хаосу в тщательно обработанном процессе загрузки и преобразования данных.

Язык M настолько обширен, что требует целой книги. Я сознательно решил представить здесь только самое поверхностное из введения. Для получения более подробной информации я предлагаю вам обратиться к документации Microsoft.

Что такое язык M?

M является функциональным языком. Он не был создан для программирования общего назначения. Опытные программисты будут тщетно искать структуры и методы кодирования, которые являются основными для других языков.

Я предпочитаю представлять новичкам функциональный язык M тремя способами:

- M существует для выполнения простой функции – загрузки и преобразования данных.
- M существует в виде серии из одной или нескольких функций, каждая из которых преобразует набор входных значений к одному выходному значению.

- M построен на компендиуме[1] из более чем 700 встроенных функций, каждая из которых предназначена для выполнения определенного фрагмента логики загрузки или преобразования данных.

Чтобы завершить вихревое введение, вам также нужно знать, что:

- M чувствителен к регистру, поэтому вам нужно быть очень осторожным при вводе ключевых слов функций и имен переменных.
- M строго типизирован. Вы должны преобразовывать основные типы данных в соответствующий тип. M не будет делать это автоматически.
- M построен на наборе ключевых слов, операторов и знаков препинания.

M и редактор Power Query

Хорошая новость о M заключается в том, что каждый шаг в процессе загрузки и преобразования данных в графическом интерфейсе редактора Power Query, создавал для вас M-код – автоматически. Из этого следует, что:

- Вам не обязательно начинать писать код M с чистого листа. Часто можно использовать интерфейс редактора запросов для выполнения большей части работы, а затем настроить автоматически созданный код, чтобы добавить пользовательские элементы.
- Вам не нужно изучать более 700 функций для написания кода M, так как редактор запросов может найти и написать для вас многие из соответствующих инструкций.
- Интерфейс редактора запросов тесно связан с тем, как написан код M. Таким образом, понимание того, как использовать интерфейс, поможет вам понять, что такое код M и как он работает.

Изменение кода для шага

Вы можете отредактировать код, который создается автоматически при каждом добавлении или изменении шага запроса. Скачайте приложенный файл BrilliantBritishCars.xlsx. Откройте новый файл Excel. Пройдите *Данные –> Получить данные –> Из файла –> Из книги*. Выберите BrilliantBritishCars.xlsx. В нем единственная таблица *BaseData*. Выберите её и кликните *Преобразовать данные*. Откроется редактор Power Query. Выберите столбец *IsDealer* и удалите его. Видите ли вы строку формул? Если нет, перейдите на вкладку *Просмотр* и поставьте галочку *Строка формул*. Вы увидите код M в строке формул:

Рис. 1. Код M для примененного шага

В строке формул замените *IsDealer* на *ReportingYear*. Нажмите *Enter* или щелкните галочку в строке формул, чтобы подтвердить изменения. В окне предварительного просмотра вы снова увидите столбец *IsDealer*, зато будет удален столбец *ReportingYear*. Вы также могли бы сделать это с помощью интерфейса, щелкнув значок шестеренки в списке *Примененные шаги* для этого шага.

При изменении кода M для шага запроса в строке формул следует помнить, что:

- Любая ошибка приведет не только к сбою шага, но и к сбою всего процесса загрузки и преобразования данных, начиная с текущего шага.

---

[1] *Здесь и далее примечания Багузина, подготовленные совместно с ChatGPT*. Я не знал, что такое компендиум и спросил ChatGPT что означает термин. Чтобы проверить, как работает ChatGPT, я повторил вопрос. Привожу оба ответа без редактуры.
Компендиум – это сборник или обзор, содержащий краткую информацию о различных темах или предметах. Это может быть как книга, содержащая краткие описания терминов или идей, так и справочник, в котором собрана информация о конкретной области знаний или предмета. В целом, компендиум является сокращенным и упорядоченным источником информации, который может использоваться для быстрого ознакомления с темой или для получения общей представления о предмете.
Компендиум – это сборник, обзор или сводный документ, который содержит информацию о какой-то теме или предмете. Обычно компендиум является сжатой версией более длинного и подробного материала, который может быть использован в качестве быстрого справочного руководства или сокращенной версии для изучения определенной темы. Компендиумы могут быть связаны с различными областями знаний, такими как медицина, финансы, юриспруденция, технологии и т.д.

- Вы должны помнить, что M чувствителен к регистру, и даже малейшая ошибка на этом уровне может привести к сбою всего процесса.
- Использование кавычек для определения литеральных элементов (таких как имена столбцов) должно соблюдаться.
- M щедро использует обычные, квадратные и фигурные скобки. Может потребоваться некоторая практика и понимание лежащей в их основе логики, чтобы полностью оценить использование скобок в различных контекстах.

К счастью, M будет предоставлять довольно четкие сообщения об ошибках. Например, если вы введете ошибочное имя поля, вы увидите сообщение:

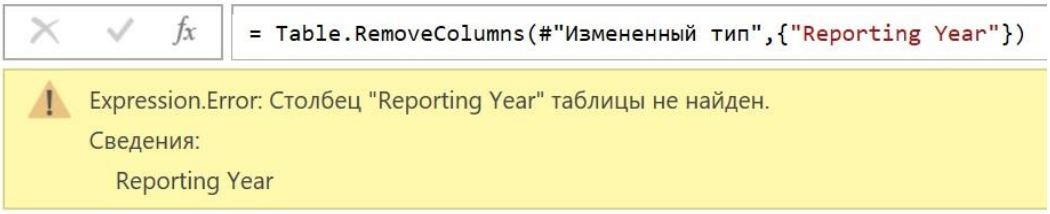


Рис. 2. Сообщение об ошибке

Вы можете удалить два столбца, а не один:

= Table.RemoveColumns(#"Измененный тип",{"ReportingYear", "IsDealer"})

Если вы хотите написать код M, проще начать с существующих шагов или создать «фиктивный» шаг, используя функцию интерфейса. Вы получить представление о том, как выглядит код M для конкретной функции.

M выражения

Ниже показана основная структура шага. Однако только интерфейс Power Query называет это шагом. В M это называется выражением.

Рис. 3. Выражение M

Каждое выражение M состоит из функций. Это может быть одна из встроенных функций (например, используемая здесь функция Table.RemoveColumns) или пользовательская функция. Также могут использоваться вычисления или логические выражения.

Код M — это серия отдельных действий (или шагов, как их называет интерфейс Power Query). Эти действия связаны в «цепочку», где каждое выражение построено на предшествующем выражении и ссылается на него. На рисунке 3 выражение ссылается на предшествующее выражение #"Измененный тип".

M-выражения могут стать чрезвычайно сложными и включать в себя несколько функций, как сложные формулы Excel.

Написание M путем добавления пользовательских столбцов

Другим способом написания M-кода является добавление пользовательских (или вычисляемых) столбцов. Они могут делать много вещей, в частности:

- Объединение (или соединение) существующих столбцов
- Добавление расчетов в таблицу данных
- Извлечение определенной части столбца
- Добавление флагов в таблицу на основе существующих данных

Давайте объединим столбцы *Make* и *Model* с пробелом между ними. Продолжим предыдущий пример. В редакторе Power Query перейдите на вкладку *Добавление столбца.* Щелкните

Настраиваемый столбец. Откроется диалоговое окно *Настраиваемый столбец*. Выберите столбец *Make* в списке столбцов справа, нажмите кнопку *Вставить*. В поле *Настраиваемая формула столбца* появится =[Make]:

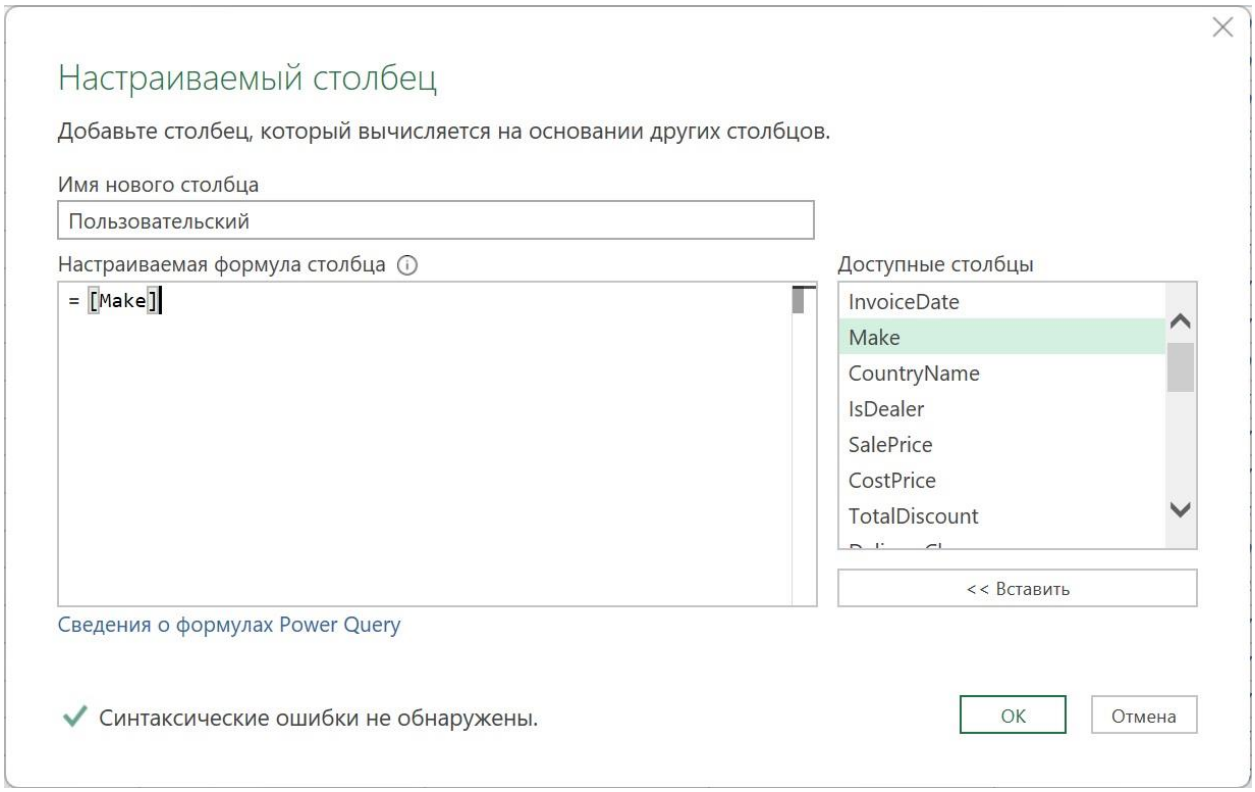


Рис. 4. Диалоговое окно *Настраиваемый столбец*

После =[Make] введите &" "&. Щелкните столбец *Model* в списке столбцов справа и нажмите кнопку *Вставить*. Поместите курсор в поле *Имя нового столбца* и введите имя столбца *CarType*. Формула примет финальный вид:

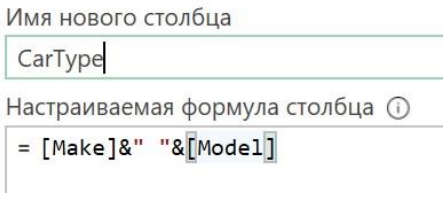


Рис. 5. Формула настраиваемого столбца

Нажмите Ok. Новый столбец добавляется крайним справа в таблице данных. Он содержит результаты формулы. Вставленный столбец отображается в списке *Примененные шаги*. Строка формул содержит:

Рис. 6. Формула настраиваемого столбца в строке формул

Всегда следует заключать имя столбца в квадратные скобки.

Строка M-кода следует принципам показанным на рис. 3. Есть M-функция (Table.AddColumn), предыдущее выражение (#"Измененный тип"), и код выражения, в данном случае добавляющий новый столбец. Ключевое слово *each* является соглашением M, указывающим, что к каждой записи в столбце будет применена формула.

Расширенный редактор

Чаще код M пишут не в строке формул, а в *расширенном редакторе* Power Query. И вот почему:

- Расширенный редактор показывает все выражения, входящие в M-запрос.
- Это значительно облегчает понимание последовательности выражений.

- Расширенный редактор имеет встроенный *IntelliSense*. Это означает, что вы можете видеть функции и константы M при вводе начальных символов.
- Он имеет синтаксическую проверку, которая показывает место, где обнаружена синтаксическая ошибка.
- Он использует цвета, чтобы сделать код M более читаемым и понятным.

Выражения в расширенном редакторе

Выражения M, которые можно увидеть по отдельности в строке формул, не существуют в вакууме. Напротив, они всегда являются частью согласованной последовательности событий загрузки, очистки и преобразования данных. Давайте взглянем на блок кода M, который был создан нашими действиями выше. Чтобы увидеть код M, в редакторе Power Query перейдите на вкладку *Главная* и кликните кнопку *Расширенный редактор*. Откроется окно *Расширенный редактор*:

Рис. 7. Блок кода M в расширенном редакторе

Это диалоговое окно содержит всю структуру созданного процесса подключения и преобразования. Он содержит следующие основные элементы:

- Последовательность выражений (которые являются шагами)
- Выражение *let*, которое действует как начало внешнего контейнера для последовательности выражений преобразования данных.
- Выражение *in*, которое завершает внешний контейнер, и возвращает выходные данные всего запроса.
- Каждое выражение имеет имя, и его имя можно увидеть в списке *Примененные шаги*.
- Каждое выражение ссылается на другое выражение (почти всегда на предыдущее), кроме первого.
- Все выражения, кроме конечного, заканчиваются запятой.
- Выражение может выполняться в нескольких строках кода.

Выражение let

Оператор let является ключевым элементом языка M. Он существует, чтобы позволить набору выражений вернуть одно значение. Внутри блока let…in каждому выражению присваивается имя переменной. Эти переменные образуют структурированную последовательность процессов вычисления, которые затем используются в выходном выражении, следующем за инструкцией in. Вы можете считать его «единицей обработки» во многих отношениях. Операторы *let* могут быть вложены для повышения гибкости.

В большинстве операторов let последовательность переменных будет упорядочена сверху вниз. Каждое именованное выражение ссылается на предыдущее и идет за ним. Именно так редактор запросов представляет именованные выражения в виде шагов и, как правило, является самым простым способом написания M. Однако технически нет необходимости упорядочивать выражения, они могут идти в любом порядке.

Изменение кода M в расширенном редакторе

Рассмотрим пример. Добавьте новый запрос, который подключается к базе данных SQL Server. Я используя экземпляр SQL и базу данных на моем ПК. Выберите запрос. Откройте *Расширенный редактор*.

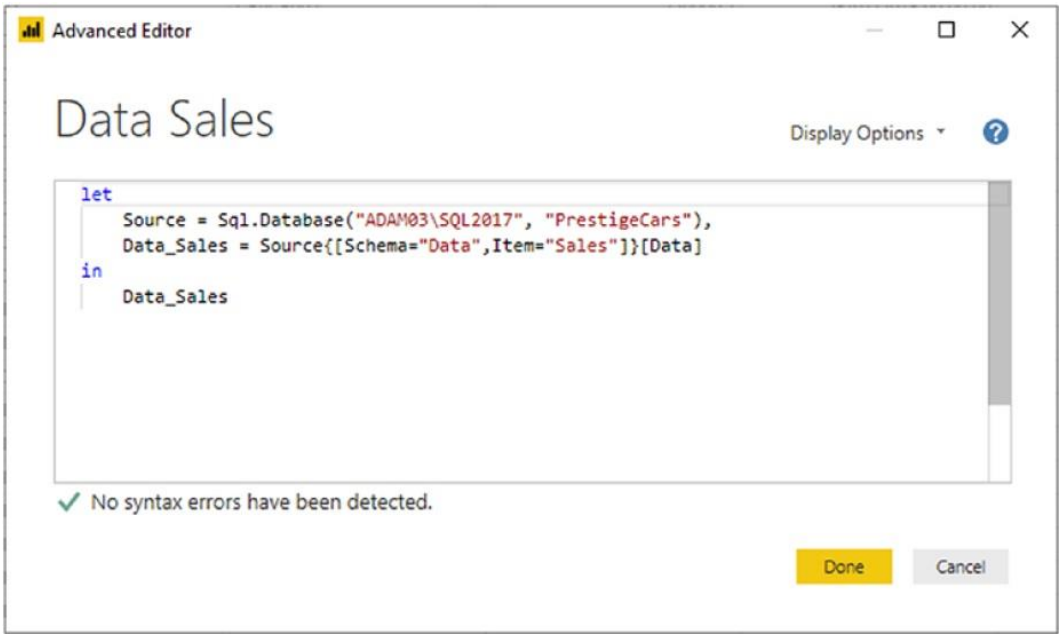


Рис. 8. Расширенный редактор для изменения подключения к базе данных

Можно изменить любой из следующих элементов: имя сервера в строке Source (в настоящее время "ADAM03\SQL2017"), имя базы данных во втором параметре (в настоящее время Name="PrestigeCars"), имя схемы (в настоящее время Schema="Data"), имя таблицы (в настоящее время Item="Sales"). Нажмите кнопку *Done (Готово)*.

Этот подход работает, не блокируя изменения кода, даже если вы введете неверные данные. Можно нажать кнопку *Cancel (Отмена)*, чтобы проигнорировать любые изменения, внесенные в код M в расширенном редакторе. Редактор запросов попросит вас подтвердить, что вы действительно хотите отменить изменения.

Проверка синтаксиса

Предположим, что вы допустили ошибку в коде. Внизу вы видите сообщение об ошибке, и все ошибки будут подчеркнуты красным цветом. При нажатии на ссылку *Показать ошибку* будет выделен источник ошибки на сером фоне. Я удалил запятую перед *each*.

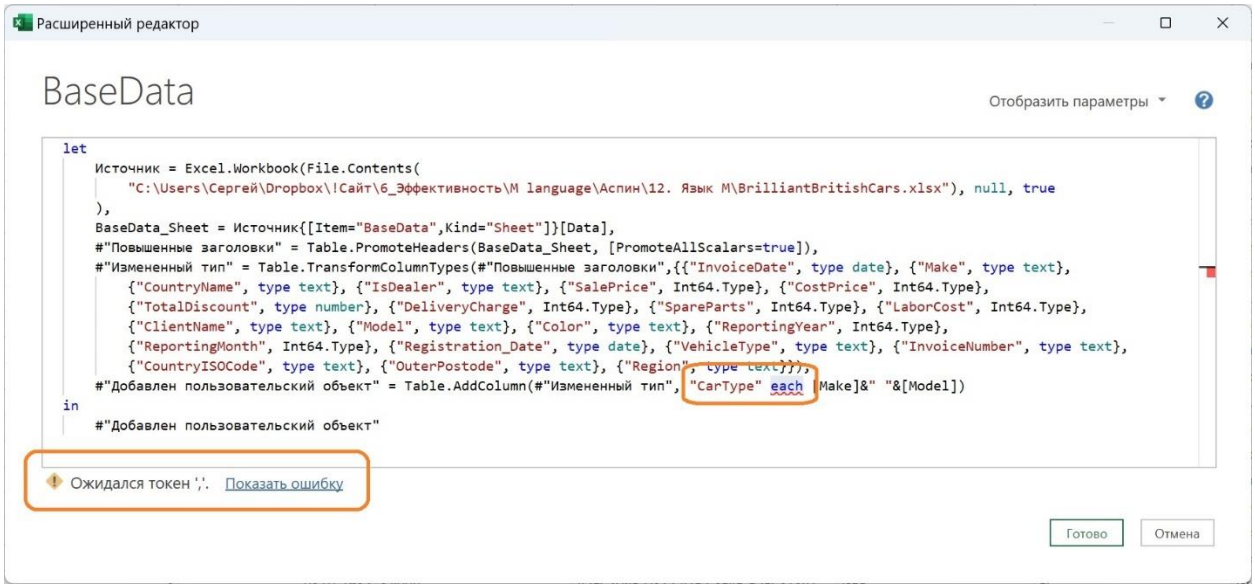


Рис. 9. Проверка синтаксиса в расширенном редакторе

Параметры расширенного редактора

В правом верхнем углу окна *Расширенный редактор* нажмите всплывающий треугольник справа от «Параметров отображения»:

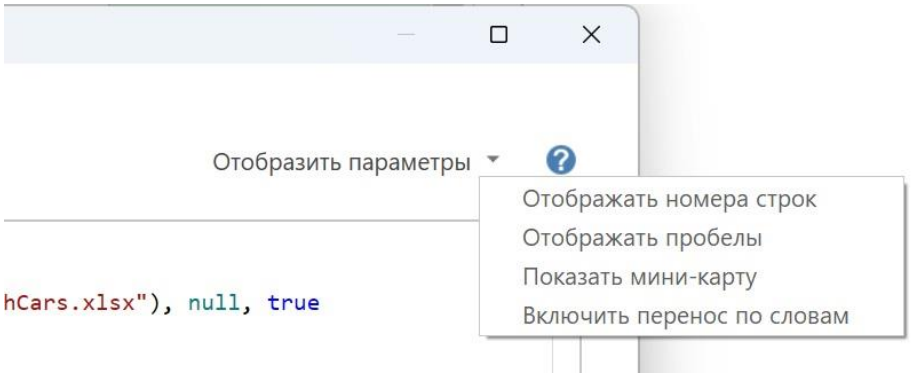


Рис. 10. Параметры расширенного редактора

Выберите *Включить перенос по словам*. Слишком длинные выражения M будут разбиты на несколько строк, чтобы поместиться на видимой части экрана. *Отображать номера строк* – добавляет номера строк слева от кода. *Отображать пробелы* – пробелы в виде серых точек появятся от левой границы до первого символа в каждой строке. *Показать мини-карту* – показывает обзор структуры кода справа от расширенного редактора.[2]

Основные функции M

Язык M обширен — слишком обширен, поэтому в одной главе можно представить лишь беглый обзор. В M есть несколько ключевых категорий функций. В том числе: текстовые, даты и времени, логические и числовые. Я ограничился этим неполным списком лишь потому, что функции входящие в него легче понять и начать использовать.

Некоторые функции M имеют аналоги в Excel. Например, вы можете модифицировать формулу на рис. 6, чтобы извлечь только три левых символа из *Make*:

= Table.AddColumn(#"Измененный тип", "CarType", each Text.Start([Make],3)&" "&[Model])

Некоторые наиболее полезные текстовые функции я собрал в таблице:

[2] У меня картинка была очень мелкой((

Output	Code Snippet	Description
Left	`Text.Start([Make],3)`	Returns the first three characters from the Make column.
Right	`Text.End([Make],3)`	Returns the last three characters from the Make column.
Up to a specific character	`Text.Start([Make], Text.PositionOf([Make]," "))`	Returns the leftmost characters up to the first space.
Up to a delimiter	`Text.BeforeDelimiter( [InvoiceNumber], "-" ,"2")`	Returns the text before the third hyphen.
Text length	`Text.Length([Make])`	Finds the length of a text.
Extract a substring	`Text.Range([Make], 2, 3)`	Extracts a specific number of characters from a text—starting at a specified position.
Remove a subtext	`Text.RemoveRange([Make], 2, 3)`	Removes a specific number of characters from a text—starting at a specified position.
Replace a text	`Text.Replace([Make], "o", "a")`	Replaces all the o characters with an a in the text or column.
Trim spaces	`Text.Trim([Make])`	Removes leading and trailing spaces in the text or column.
Convert to uppercase	`Text.Upper([Make])`	Converts the text or column to uppercase.
Convert to lowercase	`Text.Lower([Make])`	Converts the text or column to lowercase.
Add initial capitals	`Text.Proper([Make])`	Adds initial capitals to each word of the text or column.

Рис. 11. Примеры текстовых функций

Индексы в M отсчитываются от нуля. Например, чтобы задать третий дефис, вы использовали 2, а не 3.

Некоторые операции можно выполнять как на этапе загрузки с помощью M Power Query, так и в модели данных с использованием формул на языке DAX.

С перечнем всех текстовых функций можно ознакомиться на сайте Microsoft. Всего их 45.

Числовые функции

Вот некоторые числовые функции. Я остановился на преобразовании типов, и расчетах.

Output	Code Snippet	Description
Returns an 8-bit integer	`Int8.From("25")`	Converts the text or number to an 8-bit integer.
Returns a 16-bit integer	`Int16.From("2500")`	Converts the text or number to a 16-bit integer.
Returns a 32-bit integer	`Int32.From("250,000")`	Converts the text or number to a 32-bit integer.
Returns a 64-bit integer	`Int64.From("2500000000")`	Converts the text or number to a 64-bit integer.
Returns a decimal number	`Decimal.From("2500")`	Converts the text or number to a decimal.
Returns a Double number value from the given value	`Double.From("2500")`	Converts the text or number to a floating point number.
Takes a text as the source and converts to a numeric value	`Number.FromText("2500")`	Converts the text to a number.
Rounds a number	`Number.Round(5000, 0)`	Rounds the number up or down to the number of decimals (or tens, hundreds, etc., for negative parameters).
Rounds a number up	`Number.RoundUp(5020, -2)`	Rounds the number up to the number of decimals (or tens, hundreds, etc., for negative parameters).
Rounds a number down	`Number.RoundDown(100.01235, 2)`	Rounds the number down to the number of decimals (or tens, hundreds, etc., for negative parameters).
Removes the sign	`Number.Abs(-50)`	Returns the absolute value of the number.
Raises to a power	`Number.Power(10, 4)`	Returns the value of the first parameter to the power of the second.
Modulo	`Number.Mod(5, 2)`	Returns the remainder resulting from the integer division of number by divisor.
Indicates the sign of a number	`Number.Sign(-1)`	Returns 1 if the number is a positive number, -1 if it is a negative number, and 0 if it is zero.
Gives the square root	`Number.Sqrt(4)`	Returns the square root of the number.

Рис. 12. Примеры числовых функций

Функции похожи на аналогичные в Excel и Power Pivot, за исключением того, что здесь (как и в Power Pivot) вы используете имена столбцов, а не ссылки на ячейки. С перечнем всех числовых функций можно ознакомиться на сайте Microsoft.

Функции даты

Некоторый полезный перечень функций даты.

Output	Code Snippet	Description
Day	`Date.Day(Date.FromText("25/07/2020"))`	Returns the number of the day of the week from a date.
Month	`Date.Month(Date.FromText("25/07/2020"))`	Returns the number of the month from a date.
Year	`Date.Year(Date.FromText("25/07/2020"))`	Returns the year from a date.
Day of week	`Date.DayOfWeek(Date.FromText("25/07/2020"))`	Returns the day of the week from a date.
Name of weekday	`Date.DayOfWeekName(Date.FromText("25/07/2020"))`	Returns the weekday name from a date.
First day of month	`Date.StartOfMonth(Date.FromText("25/07/2020"))`	Returns the first day of the month from a date.
Last day of month	`Date.EndOfMonth(Date.FromText("25/07/2020"))`	Returns the last day of the month from a date.
First day of year	`Date.StartOfYear(Date.FromText("25/07/2020"))`	Returns the first day of the year from a date.
Last day of year	`Date.EndOfYear(Date.FromText("25/07/2020"))`	Returns the last day of the year from a date.
Day of year	`Date.DayOfYear(Date.FromText("25/07/2020"))`	Returns the day of the year from a date.
Week of year	`Date.WeekOfYear(Date.FromText("25/07/2020"))`	Returns the week of the year from a date.
Quarter	`Date.QuarterOfYear(Date.FromText("25/07/2020"))`	Returns the number of the quarter from a date.
First day of quarter	`Date.StartOfQuarter(Date.FromText("25/07/2020"))`	Returns the first day of the quarter from a date.
Last day of quarter	`Date.EndOfQuarter(Date.FromText("25/07/2020"))`	Returns the last day of the quarter from a date.

Рис. 13. Примеры функций даты

Все функции даты доступны на сайте.[3]

Функции времени

Output	Code Snippet	Description
Hour	`Time.Hour(#time(14, 30, 00))`	Returns the hour from a time.
Minute	`Time.Minute(#time(14, 30, 00))`	Returns the minute from a time.
Second	`Time.Second(#time(14, 30, 00))`	Returns the second from a time.
Time from fraction	`Time.From(0.5)`	Returns the time from a fraction of the day.

Рис. 14. Примеры функций времени

Все функции времени доступные на сайте.

[3] Перевод этого класса функций содержит ошибку. На сайте Microsoft они называются *Функции данных*.

В M также есть функции даты и времени и функции для работы с датами, временем и часовыми поясами.

Функции длительности

M также может извлекать длительности — в днях, часах, минутах и секундах.

Output	Code Snippet	Description
Days	`Duration.Days(#duration(10, 15, 55, 20))`	Duration in days.
Hours	`Duration.Hours(#duration(10, 15, 55, 20))`	Duration in hours.
Minutes	`Duration.Minutes(#duration(10, 15, 55, 20))`	Duration in minutes.
Seconds	`Duration.Seconds(#duration(10, 15, 55, 20))`	Duration in seconds.

Рис. 15. Примеры функций длительности

Концепции языка M

Мы перейдем от начальных функций, которые можно использовать для изменения содержимого данных, к созданию и изменению самих структур данных. M ориентирован на загрузку и представление табличных структур данных. Мы рассмотрим три основные структуры данных. В совокупности они классифицируются как структурированные значения, в отличие от примитивных значений, таких как текст, число или дата. Это списки, записи и таблицы.

Язык M имеет два фундаментальных аспекта: типы данных и значения M (также называемые переменными или идентификаторами).

Типы данных M

Примитивные значения могут иметь один из следующих типов:

- Number
- Text
- Date
- Time
- DateTime
- DateTimeZone
- Duration
- Logical (Boolean)
- Binary
- Null

Существуют и другие типы, такие как function, any, or anynonnull, но мы не будем рассматривать.

Все типы данных должны быть введены определенным образом.

Data Type	Code Snippet	Comments
Number	100 0.12345 2.4125E8	Do not use formatting such as thousands separators or monetary symbols.
Text	"Calidra Power BI Training"	Always enclose in double quotes. Use two double quotes to enter the actual quotes text.
Date	#date(2020,12,25)	Dates must be year, month, and day in the #date() function.
Time	#time(15,55,20)	Times must be hour, minute, and second in the #time() function.
DateTime	#datetime(2020,12, 25,15,55,20)	Datetimes must be year, month, day, hour, minute, and second in the #datetime() function.
DateTimeZone	#datetimezone(2020,12, 25,15,55,20,-5,-30)	Datetimezones must be year, month, day, hour, minute, second, day offset, and hour offset in the #datetimezone() function.
Duration	#duration(0,1,0,0)	Days, hours, minutes, and seconds comma-separated inside the #duration() function.
Logical	true	true or false in lowercase

Рис. 16. Корректный ввод различных типов данных

Значения M

Значения M:

- Значения являются выходными данными выражений.
- Значения также являются переменными.
- Имена значений чувствительны к регистру.
- Если имя значения содержит пробелы или специальные символы, они должны быть обернуты в #"".

Интерфейс редактора Power Query делает шаги (которые являются значениями, возвращаемыми выражением) более читабельными, добавляя пробелы, где это возможно. Следовательно, эти значения всегда отображаются в коде M как #"Имя шага".

Определение собственных переменных в M

Поскольку значения, возвращаемые любым выражением, также являются переменными, из этого следует, что определение собственных переменных в M потрясающе просто. Все, что вам нужно сделать, это ввести имя переменной (с решеткой и в кавычках, если оно содержит пробелы или спецсимволы), знак равенства и определение переменной.

Следующий код определяет три параметра для функции List.Numbers(),а затем использует переменные внутри функции:

```
let
   StartNumber = 0,
   Increment = 5,
   StopValue = 100,
  Source = List.Numbers(StartNumber, StopValue, Increment)
in
   Source
```

Написание M-запросов

Предположим, что вам нужна среда для изучения примеров в оставшейся части этой главы:

1. Откройте новый файл Excel.
2. Пройдите *Данные –> Получить данные –> Из других источников –> Пустой запрос*. Откроется редактор Power Query.
3. Щелкните *Расширенный редактор*. Откроется окно *Расширенного редактора* содержащее заготовку для выражения let.

Предложение let не является обязательным в M. Вы можете просто ввести выражение. Однако я предпочитаю писать код M по-книжному, по крайней мере, для начала.

Списки

Списки – просто ряд значений. Списки могут использоваться непосредственно в модели данных. Однако они чаще используются в качестве промежуточных этапов в более сложных процессах преобразования данных. Если у вас есть опыт программирования, вам может быть полезно рассматривать списки как нечто похожее на массивы.

Список – это разделенный запятыми набор значений, заключенных в фигурные скобки, например:

{1,2,3}

После интеграции в структуру запроса M он может выглядеть так:

```
let
   Source = {1,2,3}
in
   Source
```

Если вы нажмете *Готово* M отобразит список. Вы можете создавать пользовательские списки, например, для параметров.

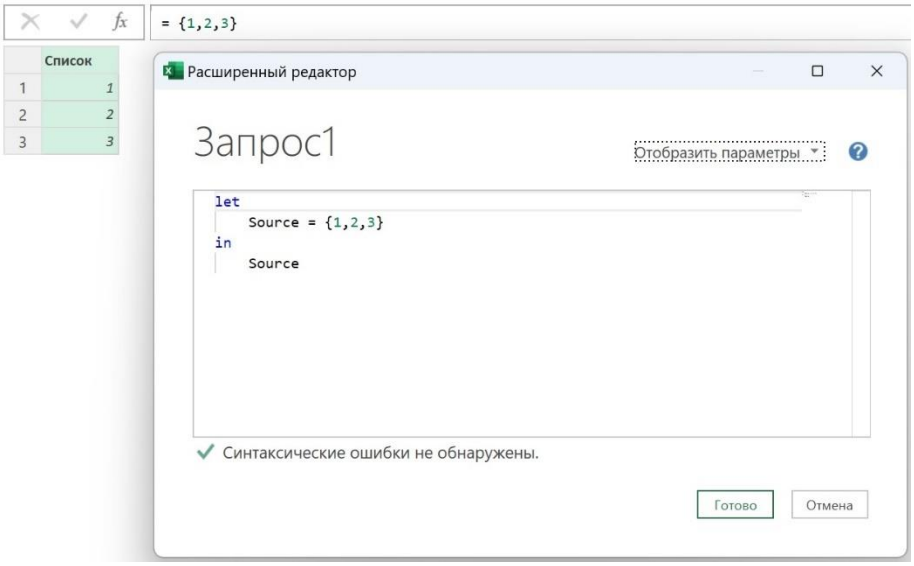


Рис. 17. Простой список

Списки:

- не ограничены по размеру и могут содержать данные одного типа (т.е. все элементы числовые, даты или текст) или разных.
- могут быть пустыми {}.
- можно вводить по горизонтали или вертикали; предыдущий список можно ввести как:

```
let
   Source = {
      1,
      2,
      3
   }
in
   Source
```

Создавать списки легко. Знать, когда использовать списки – трудно.

Создание последовательностей с помощью списков

Списки имеют много применений в M, но есть одна область, где они особенно хороши – генерация последовательностей чисел, дат или текстов:

Code Snippet	Description
`{1..100}`	An uninterrupted sequence of numbers from 1 to 100, inclusive.
`{1..100, 201..400}`	An uninterrupted sequence of numbers from 1 to 100, then from 201 to 400.
`List.Numbers(0, 100, 5)`	Starting at zero increments by 5 until 100 is reached.
`{"A".."Z"}`	The uppercase letters A through Z.
`List.Dates(#date(2020, 1, 1), 366, #duration(1, 0, 0, 0))`	Each individual day for the year 2020—starting on January 1, 2020, 366 days (expressed as a duration in days) are added.
`List.Times(#time(1, 0, 0), 24, #duration(0, 1, 0, 0))`	Each hour in the day starting with 1 AM.

Рис. 18. Генерация списков

Доступ к значениям из списка

В простейшем случае это делается с помощью позиционных ссылок:

```
let
   Start = {"George","Bill","George W.", "Barack", "Donald"},
   source = Start{3}
in
   source
```

Будет возвращен четвертый элемент списка – *Barack*.

Функции списка

Функций списка слишком много, чтобы подробно рассматривать их здесь. Вот самые простые:

Output	Code Snippet	Description
First value	`List.First(MyList)`	Returns the first element in a list.
Last value	`List.Last(MyList)`	Returns the last element in a list.
Sort list values	`List.Sort(MyList)`	Sorts the values in a list.
Extract range	`List.Range(MyList, 4)`	Extracts a range of values from a list.
Return value(s)	`List.Select(MyList , each _ ="Adam")`	Returns the elements from a list that match a criterion.
Generate a list	`List.Generate()`	Creates a list of sequential values.
Aggregate values	`List.Sum(MyList)`	Aggregate the numeric values in a list.
Replace values	`List.ReplaceMatchingItems (MyList, {"Joe", "Fred"})`	Replaces a range of values in a list.
Convert to list	`Table.Column(MyList)`	Returns a column from a table as a list.

Рис. 19. Функции списка

Полный перечень функций списка найдете на-сайте Microsoft.

Записи

Если списки можно рассматривать как столбцы данных, записи представляют собой строки данных. Вполне возможно, что вам придется определять записи при создании более сложных процедур преобразования данных в M. Пример простейшей записи:

```
let
```

```
    Source = [Surname = "Aspin", FirstName = "Adam"]
in
    Source
```

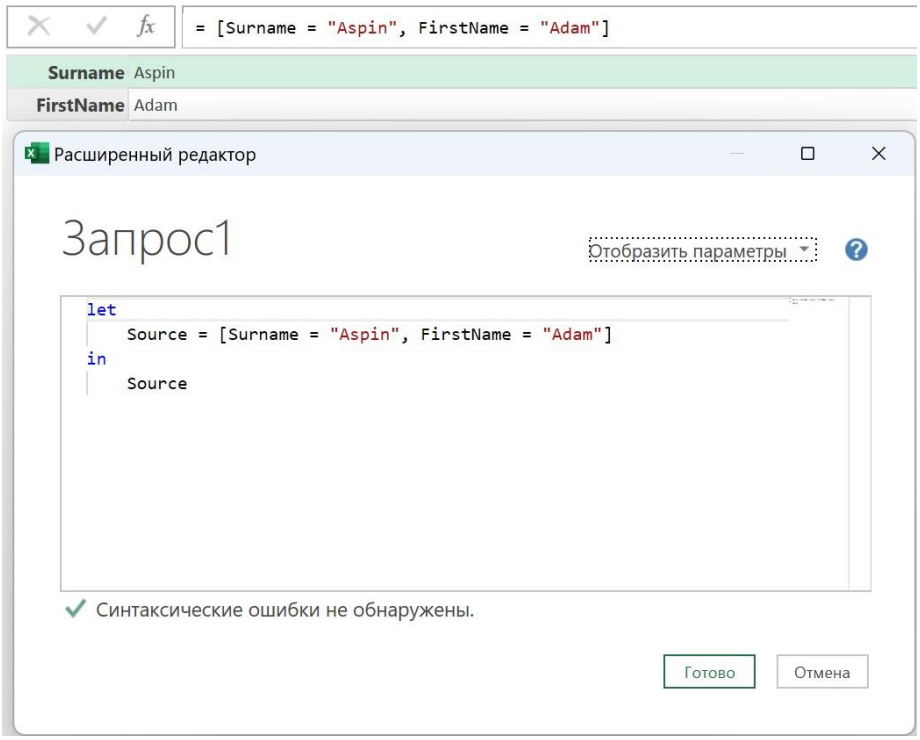


Рис. 20. Запись

Если вам нужно получить доступ к данным в записи, добавьте имя переменной записи с именем поля в квадратных скобках:

```
let
    Source = [Surname = "Aspin", FirstName = "Adam"],
    Output = Source[Surname]
in
    Output
```

Есть несколько функций записи, которые могут оказаться полезными:

Output	Code Snippet	Description
Add field	`Record.AddField()`	Adds a field to a record.
Remove field	`Record.RemoveFields()`	Removes a field from a record.
Rename fields	`Record.RenameFields()`	Renames a field in a record.
Output field	`Record.Field()`	Returns the value of the specified field in the record.
Count	`Record.FieldCount()`	Returns the number of fields in a record.

Рис. 21. Функции для работы с записями

Полный список функций для работы с записями.

Таблицы

Таблицу в M можно создать вручную с помощью функции #table(). Сначала идет набор заголовков столбцов/полей, где имя каждого поля заключено в двойные кавычки, а набор имен полей заключен в фигурные скобки. Далее идут строки данных, каждая из которых заключена в фигурные скобки и разделена запятыми, где коллекция строк также обернута фигурными скобками:

```
#table(
    {"Surname", "FirstName"},
```

```
  {
    {"Johnson","Boris"},
    {"Macron","Emmanuel"},
    {"Merkel","Angela"},
  }
)
```

Слабость этого подхода заключается в том, что не определены типы полей. Лучше использовать такой код:

```
#table(
  type table
    [
      #"Surname" = text,
      #"FirstName" = text
    ],
  {
    {"Johnson","Boris"},
    {"Macron","Emmanuel"},
    {"Merkel","Angela"},
  }
)
```

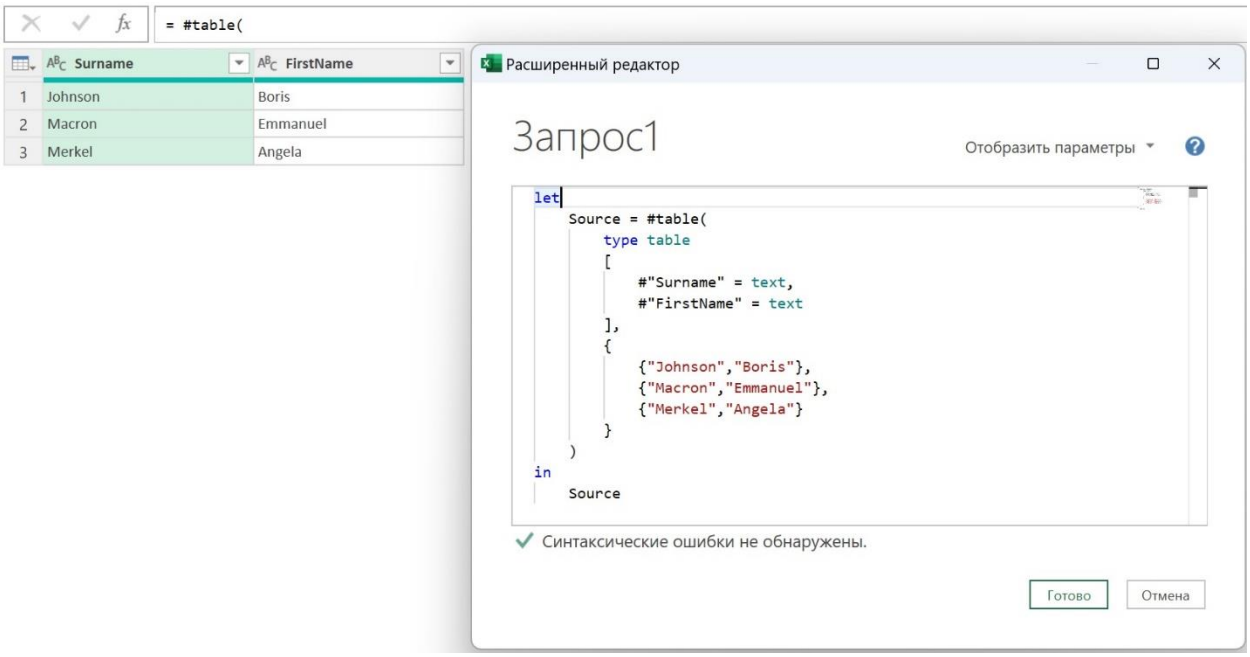


Рис. 22. Таблица с определенными типами данных

Вот некоторые табличные функции:

Output	Code Snippet	Description
Merge tables	`Table.Combine()`	Merges tables of similar or different structures.
Number of records	`Table.RowCount()`	Returns the number of records in a table.
First	`Table.First()`	Returns the first record in a table.
Last	`Table.Last()`	Returns the last record in a table.
Find rows	`Table.FindText()`	Returns the rows in the table that contain the required text.
Insert rows	`Table.InsertRows()`	Inserts rows in a table.
Output rows	`Table.Range()`	Outputs selected rows.
Delete rows	`Table.DeleteRows()`	Deletes rows in a table.
Select columns	`Table.SelectColumns()`	Outputs selected columns.

Рис. 23. Табличные функции

Полный перечень табличных функций.

Другие функциональные области

Мы сделали краткий обзор некоторых основных концепций и функций, но есть многое, что еще предстоит узнать, если вы хотите освоить M. Если вы действительно заинтересованы в получении дополнительной информации, я предлагаю вам поискать в документации Microsoft элементы, описанные следующей в таблице, чтобы расширить свои знания.

Function Area	Description
Accessing data functions	Access data and return table values.
Binary functions	Access binary data.
Combiner functions	Used by other library functions that merge values to apply row-by-row logic.
DateTime functions	Functions applied to datetime data.
DateTimeZone functions	Functions applied to datetime data with timezone information.
Expression functions	M code that was used for expressions.
Line functions	Convert data to lists of values.
Replacer	Used by other functions in the library to replace a given value in a structure.
Splitter	Splits values into sub-elements.
Type	Returns M types.
Uri	Handles URLs and URIs.
Value	Handles M values.

Рис. 24. Другие функциональные области

Пользовательские функции в M

M также позволяет писать пользовательские функции, которые могут многократно выполнять специфические задачи. В качестве примера простой пользовательской функции попробуйте добавить следующий код в пустой запрос:

```
let DiscountAnalysis =
   (Discount as number) =>
      if Discount < 10 then
         "Poor" else "Excellent"
in
   DiscountAnalysis
```

При закрытии *Расширенного редактора* вы увидите, что этот запрос распознан как функция M:

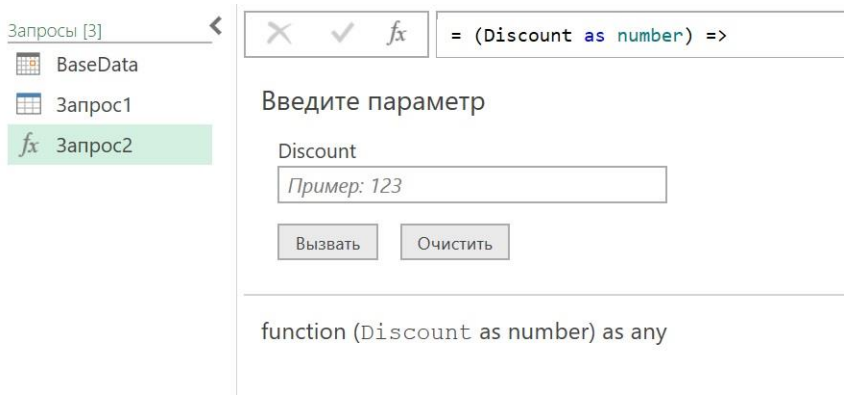


Рис. 25. Определяемые пользователем функции в M

Теперь можно вызвать функцию в интерактивном режиме, введя значение в качестве скидки и нажав кнопку *Вызвать*. Вы также можете использовать эту функцию внутри другого кода M. Именно для этого вы и создали пользовательскую функцию.

Следующий фрагмент демонстрирует более сложную функцию, которая добавляет ведущие нули к дню и месяцу, если они необходимы:

```
let
   FormatDate = (InDate as date) =>
   let
      Source =
         Text.PadStart(
            Text.From(Date.Month(InDate)),2,"0"
         )
         & "/"
         & Text.PadStart(
            Text.From(Date.Day(InDate)),2,"0"
         )
         & "/"
         &Text.From(Date.Year(InDate))
   in
      Source
in

   FormatDate
```

Добавление комментариев к коду M

Код M может быть довольно сложным. Хорошо бы вам вспомнить, почему вы создали такой код, когда вернетесь к нему через несколько недель или месяцев. Один из простых способов облегчить себе жизнь — добавить комментарии. Это можно сделать как для написанного кода, так и для запросов, созданных автоматически.

Однострочные комментарии. Чтобы прокомментировать одну строку просто добавьте две косые черты— например:

```
//This is a comment
```

Всё содержимое строки, от двух косых черт до конца строки, будет считаться комментарием.

Многострочные комментарии. Многострочные комментарии могут охватывать несколько строк или даже части строк. Они охватывают весь текст, который заключен в /* ... */.

```
/* This is a comment
Over
Several lines */
```