Язык M Power Query. Выражения и let

В Интернете довольно много примеров использования языка M Power Query для решения той или иной проблемы. Если можно просто скопировать и вставить код образца в ваш запрос, всё Ok. Но что делать, если требуется правка. Для этого нужно хоть немного разбираться в основах языка и его синтаксисе. Умники отсылают к официальной спецификации языка, но, поверьте, это не то, что нужно начинающим. Так что в этой и последующих заметках попробую объяснить азы доступным языком.[1]

> Если вы впервые сталкиваетесь с Power Query, рекомендую начать с Марк Мур. Power Query.

Самая простая запись

Откройте имеющийся или пустой запрос в расширенном редакторе, и вы увидите выражение *let*. На самом деле, вы видели *let* так часто, что могли подумать, что для языка M это обязательно. Совсем нет! Допускаются и более простые выражения.

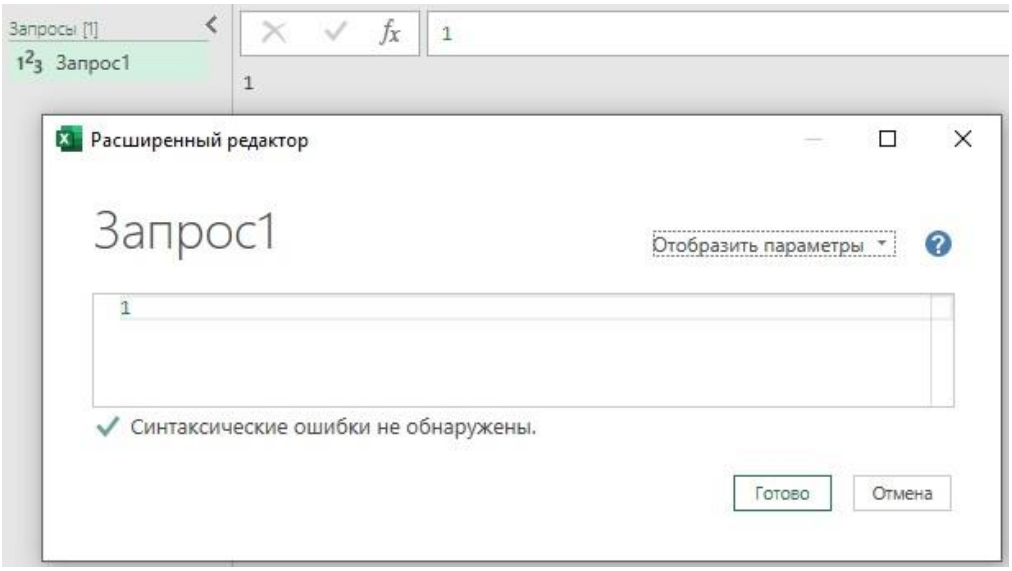

Рис. 1. Самая простая запись на языке M

Приведенный выше M-код содержит выражение, которое выдает значение, в данном случае числовое значение 1. Как и в этом простейшем примере, *let* также является выражением, которое возвращает значение. Однако *let* позволяет определять промежуточные выражения, результаты которых присваиваются переменным. Эти промежуточные выражения затем могут быть использованы для получения конечного значения, возвращаемого выражением *let*.

Разбиение большого выражения на промежуточные с присвоением каждому шагу имени облегчает чтение кода. Это также позволяет ссылаться на промежуточные шаги несколько раз по мере приближения к конечному значению выражения *let*.

[1] Это сокращенный перевод статьи Ben Gribaudo Power Query M Primer (part 1): Introduction, Simple Expressions & let.

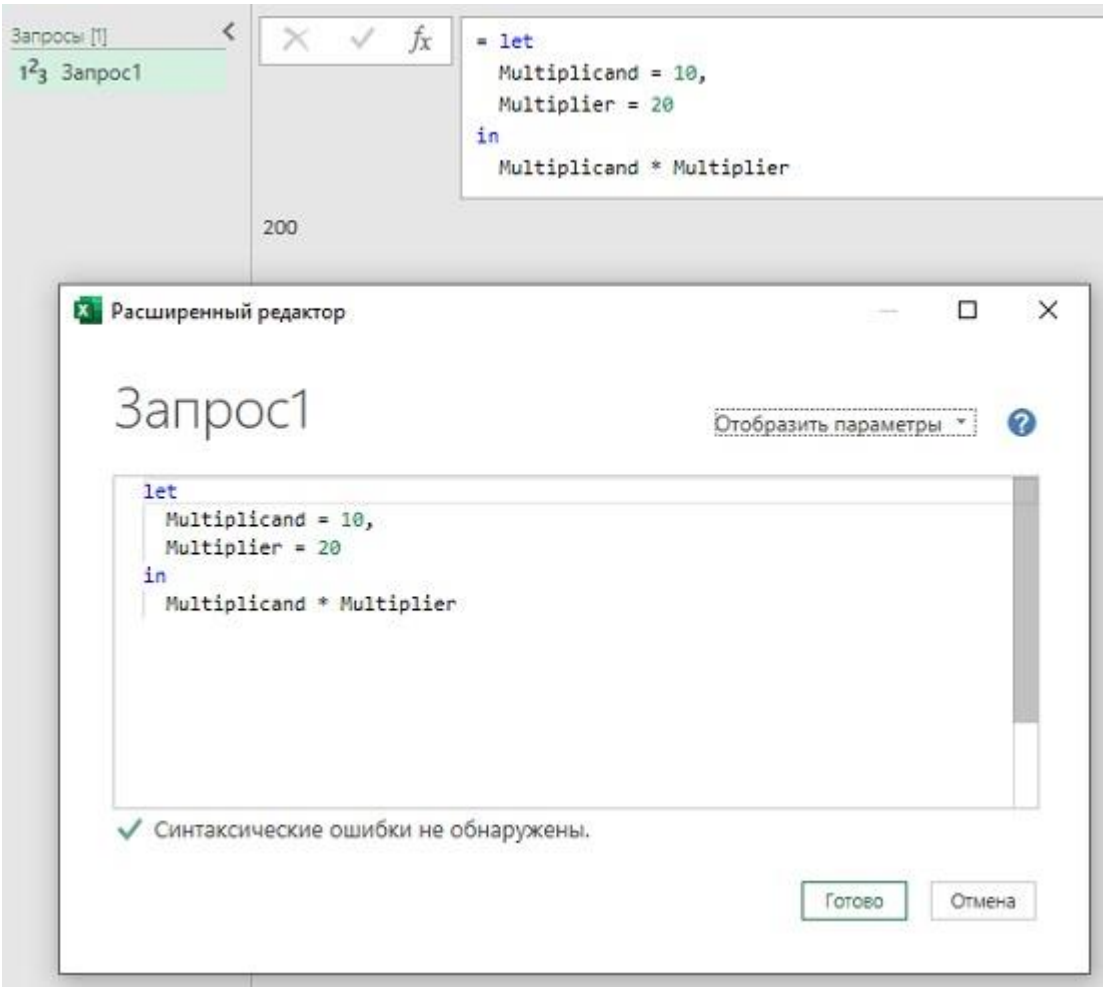

Рис. 2. Использование составного выражения *let*

В первой части этого выражения *let* переменные определяются по имени, затем знак равенства, а далее выражение, создающее значение, присваиваемое переменной. Определения переменных разделяются запятыми. Часть *in* – это выражение, которое определяет, что возвращает *let*.

Чудаки

Поскольку цель *let* – разрешить использование переменных, имеет смысл, чтобы по крайней мере одна из них использовалась в выражении *in*. Однако это не является обязательным. *in* может игнорировать все только что определенные переменные и возвращать что-то еще:

```
let
   Multiplicand = 10,
   Multiplier = 20
in
   2 + 3
```

Хм... зачем тратить усилия на определение переменных только для того, чтобы оставить их неиспользованными?! К счастью, этот стиль встречается редко. Просто знайте, что это возможно.

Единственная переменная в in

Как правило *in* состоит из одной переменной…

```
let
   Multiplicand = 10,
   Multiplier = 20,
   Result = Multiplicand * Multiplier
in
   Result
```

… а не из выражения, как на рис. 2

Хотя результат обоих выражений одинаков, стиль «одна переменная в *in*» визуализирует шаги запроса. Управляя запросом с помощью интерфейса редактора Power Query вы привыкли видеть список шагов. Каждый из этих шагов представляет собой присвоение переменной значения. Имена шагов это и есть имена переменных.

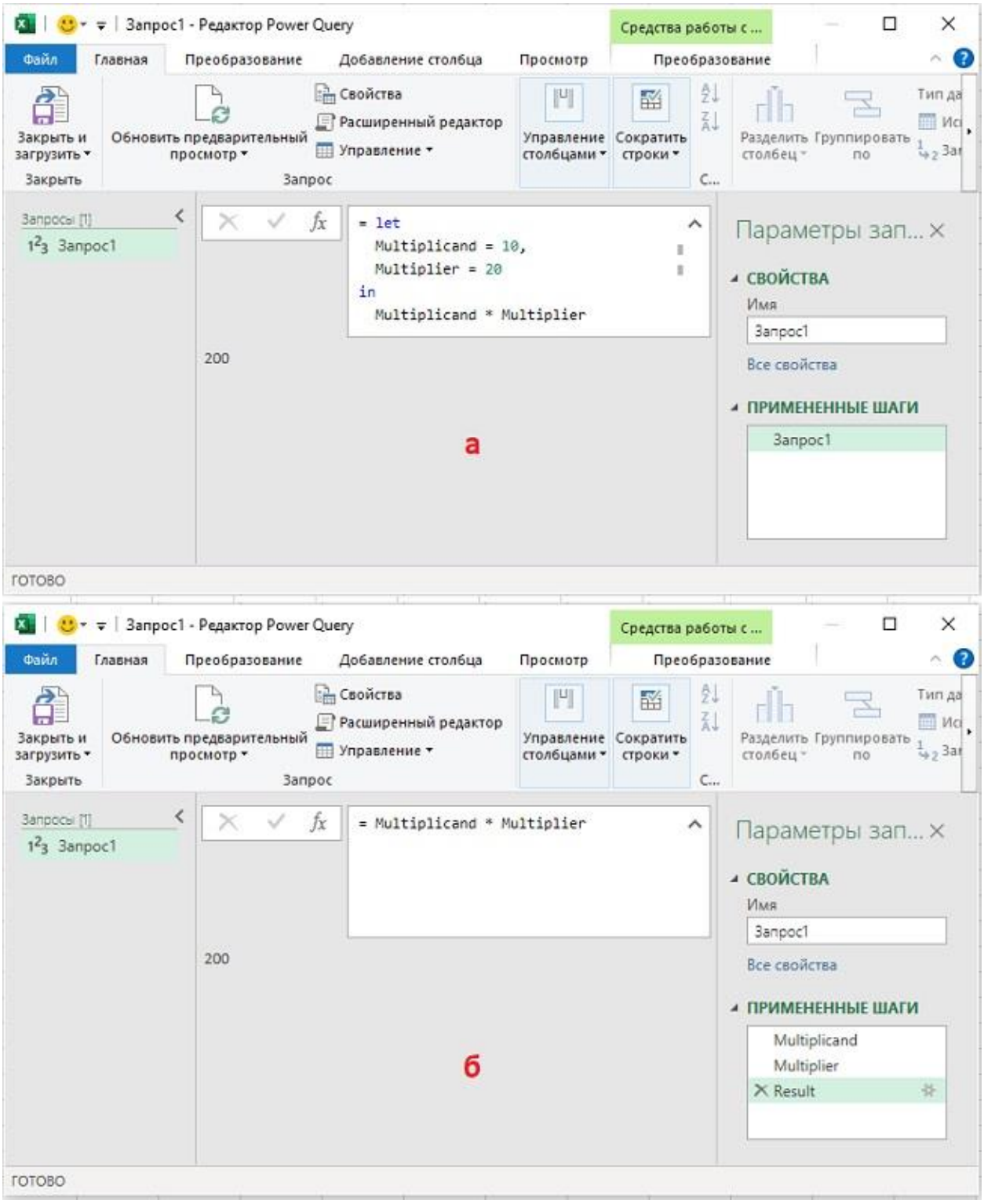

Рис. 3. Единственная переменная в *in* позволяет выделить шаги на панели *Примененные шаги*

Возможность изменения текущего шага очень полезна для целей отладки, поскольку она упрощает просмотр значения, полученного в результате промежуточного шага. Если выражение *in* более сложное, чем одна переменная, редактор запросов не свяжет его с одним шагом, и не отобразит на панели.

Вложенность

Поскольку выражение *let* – это выражение, которое выдает значение, выражения *let* можно использовать везде, где ожидаются значения. Мы можем присваивать их переменным, вкладывать их в другие выражения *let* и использовать в качестве значений для аргументов функции.

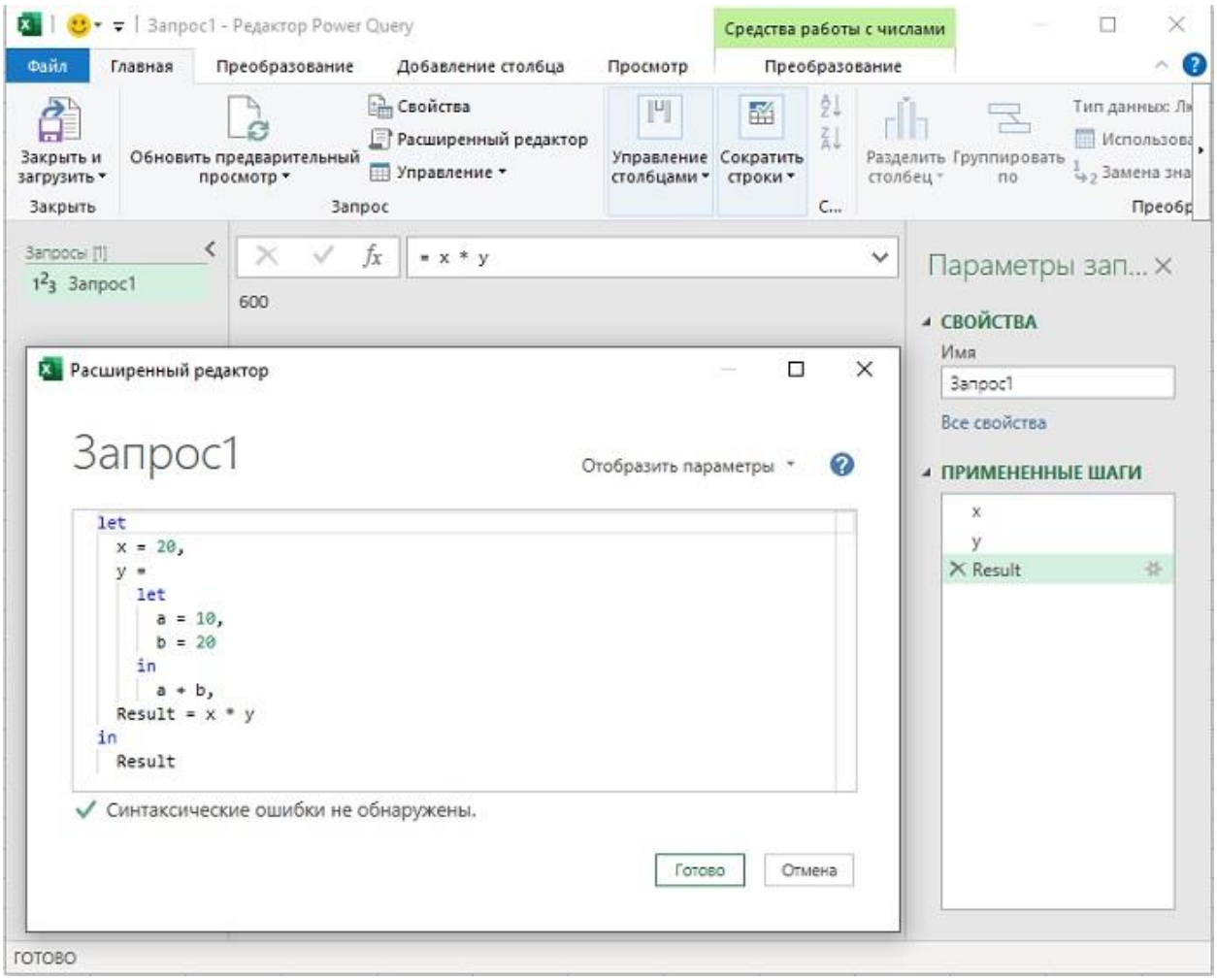

Рис. 4. Вложенные выражения *let*

Этот пример показывает, как выражение *let*, присвоенное переменной, может организовать более сложные вычисления.

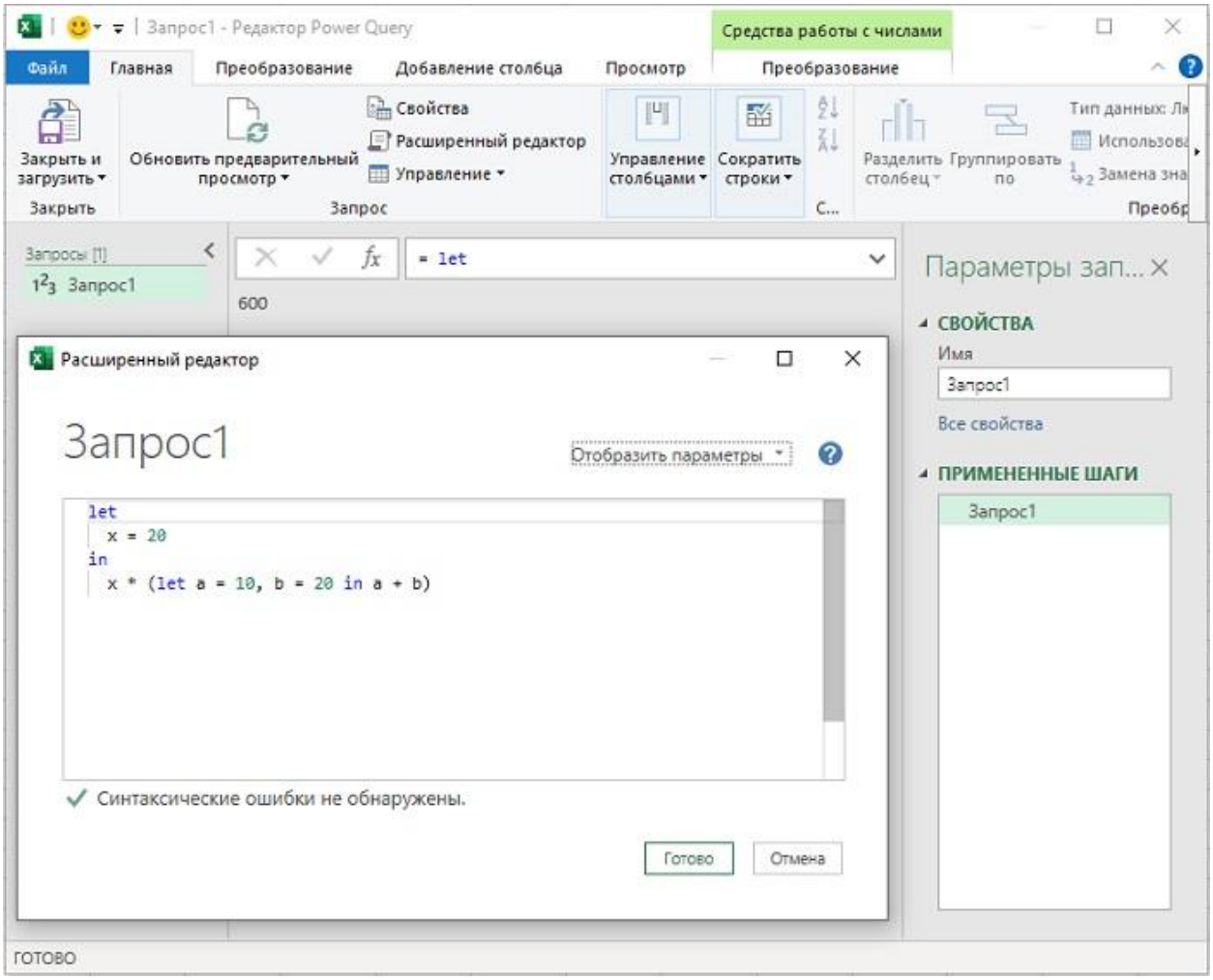

Рис. 5. Неудачная реализация

Этот пример технически допустим и даже может быть полезен для демонстрации возможностей вложенности. Но с точки зрения удобства восприятия кода, было бы лучше, если бы вложенное выражение *let* было присвоено переменной, а затем на эту переменную ссылались там, где в данный момент используется вложенное выражение.

Аналогичные замечания относятся и к следующему коду:

```
Date.AddDays(
   Date.From(DateTime.LocalNow()),
   let
      a = 2,
      b = 6
   in (a + b) * b
)
```

Лучше было бы вложенное выражение *let* присвоить переменной. Поясним, что делает этот код. Функция DateTime.LocalNow() возвращает текущую дату и время. Функция Date.From() извлекает только дату. Функция Date.AddDays() имеет два аргумента: первый – дата, второй – число дней, которые следует добавить к этой дате.