Мова формул Power Query M

- ***Типи та формати даних***
- ***Оператори мови M***
- ***Функції мови M – категорії функцій, приклади застосування***

Microsoft Power Query надає потужну функцію отримання даних, яка охоплює багато функцій.
Основною можливістю Power Query є фільтрація та об’єднання, тобто "змішані" дані з одного чи кількох із великої колекції підтримуваних джерел даних.
Центральною конструкцією в M є вираз. Вираз можна оцінити (обчислити).
Примітивне значення — це значення з однієї частини, наприклад число, логічне значення, текст або нуль. Нуль значення можна використовувати для вказівки на відсутність будь-яких даних.

Power Query M

```
123                 // A number
true                // A logical
"abc"               // A text
null                // null value
```

Значення списку — це впорядкована послідовність значень. M підтримує нескінченні списки, але якщо вони написані як літерали, списки мають фіксовану довжину. Символи фігурних дужок { і } позначають початок і кінець списку.

Power Query M

```
{123, true, "A"}    // list containing a number, a logical, and
                    //     a text
{1, 2, 3}           // list of three numbers
```

Запис — це набір полів. Поле — це пара ім’я/значення, де ім’я — це текст значення, яке є унікальним у записі поля. Синтаксис літералів для значень запису дозволяє записувати імена без лапок, ця форма також називається ідентифікатори. Нижче показано запис, що містить три поля з назвами «A», «B», і "C", які мають значення 1, 2 і 3.

Power Query M

```
[
        A = 1,
        B = 2,
        C = 3
]
```

Таблиця – це набір значень, організованих у стовпці (які ідентифікуються за назвою), і рядки. Не існує літерального синтаксису для створення таблиці, але є кілька стандартні функції, які можна використовувати для створення таблиць зі списків або записів.

Power Query M

```
#table( {"A", "B"}, { {1, 2}, {3, 4} } )
```

A	B
1	2
3	4

Записи можуть міститися в списках. За допомогою оператора позиційного індексу ({}) щоб отримати доступ до елемента в списку за його числовим індексом. Посилаються на значення в списку використовуючи індекс від початку списку від нуля. Наприклад, індекси 0 і 1 використовуються для посилання на перший і другий елементи в списку нижче:

Power Query M

```
[
    Sales =
        {
            [
                Year = 2007,
                FirstHalf = 1000,
                SecondHalf = 1100,
                Total = FirstHalf + SecondHalf // 2100
            ],
            [
                Year = 2008,
                FirstHalf = 1200,
                SecondHalf = 1300,
                Total = FirstHalf + SecondHalf // 2500
            ]
        },
    TotalSales = Sales{0}[Total] + Sales{1}[Total] // 4600
]
```

Функція — це відображення набору вхідних значень на одне вихідне значення. А функція записується, спочатку називаючи необхідний набір вхідних значень (параметрів для функція), а потім надаючи вираз, який обчислює результат функції використовуючи ці вхідні значення (тіло функції) після символу переходу до (=>).

Power Query M

```
(x) => x + 1                    // function that adds one to a value
(x, y) =>  x + y                // function that adds two values
```

Функція — це значення, як і число або текстове значення. У наступному прикладі показано а функція, яка є значенням поля Add, яке потім викликається або виконується з кількох інші поля. Під час виклику функції вказується набір значень, які відповідають логіці замінює необхідний набір вхідних значень у вираз тіла функції.

Power Query M

```
[
    Add = (x, y) => x + y,
    OnePlusOne = Add(1, 1),     // 2
    OnePlusTwo = Add(1, 2)      // 3
]
```

Метадані — це інформація про значення, пов’язане зі значенням. Метадані є представлений у вигляді значення запису, що називається записом метаданих. Поля запису метаданих можна використовувати для зберігання метаданих для значення. Кожне значення має запис метаданих. Якщо значення запису метаданих не було вказано, то запис метаданих порожній (не має полів). Записи метаданих забезпечують спосіб пов’язати додаткову інформацію з будь-яким типом значення в ненав'язливий спосіб. Зв’язування запису метаданих із значенням не змінюється значення або його поведінку. Значення запису метаданих асоціюється з існуючим значенням x за допомогою синтаксису x мета y . Наприклад, наведений нижче код пов’язує запис метаданих із рейтингом і Теги полів із текстовим значенням "Mozart" :

Power Query M

```
"Mozart" meta [ Rating = 5, Tags = {"Classical"} ]
```

Рядки стандартного числового формату використовуються для форматування типових числових типів. Стандарт рядок числового формату приймає форму [специфікатор формату][специфікатор точності] , де: Специфікатор формату - це один символ алфавіту, який визначає тип числа форматі, наприклад, валюта або відсоток. Будь-який рядок числового формату, який містить більше одного алфавітного символу, включаючи пробіл, інтерпретується як рядок спеціального числового формату.
Специфікатор точності – це необов’язкове ціле число, яке впливає на кількість цифр у результуючий рядок. Специфікатор точності контролює кількість цифр у рядку подання числа.

Валюта
123.456 ("C", en-US) -> \$123.46
123.456 ("C", fr-FR) -> 123,46 €
цілі цифри 1234 ("D") -> 1234
Експоненціальне позначення -1052.0329112756 ("E", en-US)
ціла і десяткова цифри 1234.567 ("F", en-US) -> 1234.57.
Усі числові типи - 1234.567 ("F", de-DE) -> 1234,57
Специфікатор точності: Кількість значущих цифр 123.4546 ("G4", enUS)
З плавоючою комою: вказує кількість цифр після коми -1234.56 ("F4", en-US) -> -1234.5600
Процент - 1 ("P", en-US) -> 100.00 %

Стандартний рядок числового формату можна використовувати для визначення форматування числа значення. Його можна передати в параметр формату Number.ToText.
Number.ToText(123.456, "C2") // Displays $123.46

Надати аргумент підрахунку в Text.PadStart і Text.PadEnd для визначення ширини числового поля та вирівнювання його значення за правим чи лівим краєм. Наприклад, у наведеному нижче прикладі значення валюти вирівнюється за лівим краєм у 28 символів і вирівнює значення валюти в 14-символьному полі за правим краєм.

Power Query M

```
let
    amounts = {16305.32, 18794.16},
    result = Text.Format("    Beginning Balance           Ending Balance#
(cr,lf)    #{0}#{1}",
    {
        Text.PadEnd(Number.ToText(amounts{0}, "C2"), 28),
        Text.PadStart(Number.ToText(amounts{1}, "C2"), 14)
    })
in
    result

// Displays:
//    Beginning Balance           Ending Balance
//    $16,305.32                      $18,794.16
```

Специфікатор формату "C" (або валюти) перетворює число на рядок. Специфікатор точності вказує бажану кількість десяткових знаків у рядку результату. Якщо специфікатор точності опущено, число за замовчуванням форматується як десяткове.

Якщо значення, яке потрібно відформатувати, має більше, ніж указане або за замовчуванням число десяткових чисел після коми то значення округлюється в рядку результату.

У наступному прикладі значення форматується за допомогою специфікатора формату валюти:

Power Query M

```
let
    Source =
    {
        Number.ToText(12345.6789, "C"),
        Number.ToText(12345.6789, "C3"),
        Number.ToText(12345.6789, "C3", "da-DK")
    }
in
    Source

// The example displays the following list on a system whose
// current culture is English (United States):
//       $12,345.68
//       $12,345.679
//       12.345,679 kr.
```

Специфікатор формату "D" (або десятковий) перетворює число на рядок десяткових цифр (0- 9), перед яким ставиться знак мінус, якщо число від'ємне. Специфікатор точності вказує мінімальну кількість цифр, яку буде отримано в результаті форматування. Якщо потрібно, число доповнюється нулями зліва, щоб отримати число цифри, задані специфікатором точності. Якщо специфікатор точності не вказано, за замовчуванням є мінімальне значення, необхідне для представлення цілого числа без початкових нулів.

Power Query M

```
let
    Source =
    {
        Number.ToText(12345, "D"),
        // Displays 12345

        Number.ToText(12345, "D8"),
        // Displays 00012345

        Number.ToText(-12345, "D"),
        // Displays -12345

        Number.ToText(-12345, "D8")
        // Displays -00012345
    }
in
    Source
```

Специфікатор експоненціального ("E") формату перетворює число на рядок у формі "- d.ddd…E+ddd" або "- d.ddd…e+ddd", де кожен "d" означає цифру (0-9). Рядок починається зі знака мінус, якщо число від'ємне. Специфікатор точності вказує бажану кількість цифр після коми.
Якщо специфікатор точності опущено, використовується за замовчуванням шість цифр після коми. Регістр у специфікаторі формату вказує, чи слід перед експонентою поставити "E" або "e". Експонента завжди складається зі знака плюс або мінус і мінімум трьох цифри. Експонента доповнюється нулями, щоб відповідати цьому мінімуму, якщо потрібно. На рядок результату впливає інформація про форматування поточної культури.

Power Query M

```
let
    Source =
    {
        Number.ToText(12345.6789, "E", ""),
        // Displays 1.234568E+004

         Number.ToText(12345.6789, "E10", ""),
         // Displays 1.2345678900E+004

         Number.ToText(12345.6789, "e4", ""),
         // 1.2346e+004

         Number.ToText(12345.6789, "E", "fr-FR")
         // Displays 1,234568E+004
     }
in
    Source
```

Специфікатор формату з фіксованою комою ("F") перетворює число на рядок у формі "- ddd.ddd…", де кожен "d" означає цифру (0-9). Рядок починається зі знака мінус, якщо число від'ємне. Специфікатор точності вказує бажану кількість десяткових знаків. Якщо точність специфікатор опущено, число десяткових знаків за умовчанням дорівнює 2.

Power Query M

```
let
    Source =
    {
        Number.ToText(17843, "F", ""),
        // Displays 17843.00

        Number.ToText(-29541, "F3", ""),
        // Displays -29541.000

        Number.ToText(18934.1879, "F", ""),
        // Displays 18934.19

        Number.ToText(18934.1879, "F0", ""),
        // Displays 18934


        Number.ToText(-1898300.1987, "F1", ""),
        // Displays -1898300.2

        Number.ToText(-1898300.1987, "F3", "es-ES")
        // Displays -1898300,199
    }
```

Специфікатор загального формату ("G") перетворює число на більш компактне

Numeric type	Default precision
`Byte.Type` or `Int8.Type`	3 digits
`Int16.Type`	5 digits
`Int32.Type`	10 digits
`Int64.Type`	19 digits
`Single.Type`	9 digits
`Double.Type`	17 digits
`Decimal.Type`	15 digits

Power Query M

```
let
    Source =
    {
        Number.ToText(12345.6789, "G", ""),
        // Displays 12345.6789

        Number.ToText(12345.6789, "G", "fr-FR"),
        // Displays 12345,6789

        Number.ToText(12345.6789, "G7", ""),
        // Displays 12345.68

        Number.ToText(.0000023, "G", ""),
        // Displays 2.3E-06

        Number.ToText(.0000023, "G", "fr-FR"),
        // Displays 2,3E-06

        Number.ToText(.0023, "G", ""),
        // Displays 0.0023

        Number.ToText(1234, "G2", ""),
        // Displays 1.2E+03

        Number.ToText(Number.PI, "G5", "")
        // Displays 3.1416
    }
in
    Source
```

Оператори

Мова формул Power Query M містить набір операторів, які можна використовувати. Оператори застосовуються до операндів для формування символічних виразів. Наприклад, у виразі 1 + 2 числа 1 і 2 є операндами, а оператор є оператор додавання (+). Значення оператора може змінюватися залежно від типу значень операнда.

Оператор (+)

Expression
1 + 2
#time(12,23,0) + #duration(0,0,2,0)

Оператор (&)

Function
"A" & "BC"
{1} & {2, 3}
[a = 1] & [b = 2]

Оператори для роботи з типом даних «Дата»

Operator	Left Operand	Right Operand	Meaning
x + y	time	duration	Date offset by duration
x + y	duration	time	Date offset by duration
x - y	time	duration	Date offset by negated duration
x - y	time	time	Duration between dates
x & y	date	time	Merged datetime

Типи функцій мови M

Функцій для отримання даних.

Двійкові функції - функції створюють двійкові дані та обробляють їх. Використовуються в мовах програмування, яким потрібно маніпулювати двійковими даними.

Комбінаторні функції - Ці функції використовуються іншими бібліотечними функціями, які об’єднують значення. Наприклад, Table.ToList і Table.CombineColumns застосовують функцію об’єднання до кожного рядка в а таблиці, щоб отримати одне значення для кожного рядка. корисні в контекстах, коли дані потрібно об’єднати або об’єднати.

Функції порівняння - Ці функції перевіряють рівність і визначають порядок. Використовуються під час сортування або коли потрібно порівняти елементи.

Функції дати - Ці функції для роботи з датами, часом та значення datetimezone.

Функції Обробка помилок - Ці функції можна використовувати для відстеження виняткових ситуацій.

Функції Вирази - Ці функції дозволяють побудувати та оцінити M-код. Використовуються в сценаріях, що потребують динамічного оцінювання.

Функції Значення - Ці функції створюють і викликають інші M-функції.

Логічні функції - Ці функції створюють і обробляють логічні (тобто істинні чи хибні) значення.

Типи функцій мови M (продовження)

Лінійні функції - перетворюють списки тексту в двійкові та окремі текстові.
Списки - функції створюють і обробляють значення списку.
Числові функції - Ці функції створюють і обробляють числові значення.
Функції для заміни значення - використовуються іншими функціями бібліотеки для заміни заданого значення.
Функції для розділення текстів - поширені в завданнях синтаксичного аналізу або при розподілі даних на частини.
Функції для роботи з таблицями - необхідні для взаємодії з базою даних або будь-якої обробки табличних даних.
Функції створюють і обробляють рядки запиту URI використовується у веб-розробці або будь-якій програмі, яка працює з URL-адресами

Посилання: https://powerquery.how/

функцій для отримання даних

Access.Database(database as binary, optional options as nullable record) as table Повертає структурне представлення бази даних Access. Необов'язковий запис параметр, options, можна вказати для керування такими параметрами: CreateNavigationProperties: логічне значення (true/false), яке встановлює, чи генерує властивості навігації для повернених значень (за замовчуванням false). NavigationPropertyNameGenerator: функція, яка використовується для створення імен для властивостей навігації. Параметр запису вказується, наприклад, як [параметр1 = значення1, параметр2 = значення2...].

AzureStorage.BlobContents(url as text, optional options as nullable record) as binary
Повертає вміст BLOB за URL-адресою url зі сховища Azure. Параметри функції
Розмір блоку: кількість байтів для читання перед очікуванням споживача даних. (значення за замовчуванням — 4 МБ).
RequestSize : кількість байтів, яку потрібно спробувати прочитати в одному запиті HTTP до сервера. Значення за замовчуванням становить 4 МБ.
ConcurrentRequests : Параметр ConcurrentRequests підтримує швидше завантаження даних, вказавши кількість запитів, які потрібно зробити паралельно, за ціною використання пам'яті. Потрібна пам'ять (ConcurrentRequest * RequestSize). Значення за замовчуванням – 16.

File.Contents(path as text, optional options as nullable record) as binary - Повертає вміст файлу шлях у двійковому вигляді.

Folder.Contents(path as text, optional options as nullable record) as table - Повертає таблицю, що містить рядок для кожної папки та файлу, знайденого в шляху до папки. Кожен рядок містить властивості папки або файлу та посилання на його вміст.

Функції для заміни значення

Replacer.ReplaceText(text as nullable text, old as text, new as text) as nullable text - Замінює старий текст у оригінальному тексті новим текстом. Ця функція замінника може використовувати в List.ReplaceValue і Table.ReplaceValue.

Приклад

Replacer.ReplaceText("hEllo world", "hE", "He")

Результат "Hello world«

Replacer.ReplaceValue(value as any, old as any, new as any) as any - Замінює старе значення у початковому значенні на нове значення. Ця замінна функція можна використовувати в List.ReplaceValue і Table.ReplaceValue.

Приклад Replacer.ReplaceValue(11, 11, 10)

Результат 10

Комбінаторні функції

Combiner.CombineTextByDelimiter(delimiter as text, optional quoteStyle as nullable number) as function - Повертає функцію, яка об’єднує список текстових значень в одне текстове значення за допомогою розділового знаку
Приклад Combiner.CombineTextByDelimiter(";")({"a", "b", "c"})
Результат "a;b;c«

```
let
 Source = #table(
 type table [Column1 = text, Column2 = text],
 {{"a", "b"}, {"c", "d,e,f"}}
 ),
Merged = Table.CombineColumns(
 Source,
 {"Column1", "Column2"},
 Combiner.CombineTextByDelimiter(",", QuoteStyle.Csv),
 "Merged"
 )
in
 Merged
```

Функції для розділення текстів

Splitter.SplitByNothing() as function - Повертає функцію, яка не виконує розбиття, повертаючи її аргумент як список з одним елементом.

Splitter.SplitTextByAnyDelimiter(delimiters as list, optional quoteStyle as nullable number, optional startAtEnd as nullable logical) as function - Повертає функцію, яка розділіть введені дані комою або крапкою з комою, ігноруючи лапки та роздільники починаючи з початку введення.

Приклад Splitter.SplitTextByAnyDelimiter({",", ";"}, QuoteStyle.Csv) ("a,b;""c,d;e"",f")

Результат {"a", "b", "c,d;e", "f"}

Приклад

let startAtEnd = true in

Splitter.SplitTextByAnyDelimiter({",", ";"}, QuoteStyle.Csv, startAtEnd) ("a,""b;c,d")

Результат {"a,b", "c", "d"}

Splitter.SplitTextByCharacterTransition(before as anynonnull, after as anynonnull) as function - Повертає функцію, яка розбиває текст на список тексту відповідно до переходу з одного вид характеру іншому. Параметри до і після можуть бути або списком символів або функція, яка приймає символ і повертає true/false

Приклад

Splitter.SplitTextByCharacterTransition({"A".."Z", "a.."z"}, {"0".."9"}) ("Abc123")

Результат - {"Abc", "123"}

Splitter.SplitTextByDelimiter(delimiter as text, optional quoteStyle as nullable number, optional csvStyle as nullable number) as function - Повертає функцію, яка розбиває текст на список тексту відповідно до вказаного розділювача.

Приклад Splitter.SplitTextByDelimiter(",", QuoteStyle.Csv)("a,""b,c"",d")

Результат {"a", "b,c", "d"}

Splitter.SplitTextByEachDelimiter(delimiters as list, optional quoteStyle as nullable number, optional startAtEnd as nullable logical) as function - Повертає функцію, яка розбиває текст на список тексту.
Приклад Splitter.SplitTextByEachDelimiter({",", ";"})("a,b;c,d")
Результат {"a", "b", "c,d"}
Splitter.SplitTextByLengths(lengths as list, optional startAtEnd as nullable logical) as function - Повертає функцію, яка розбиває текст на список тексту за вказаною довжиною.
Приклад Splitter.SplitTextByLengths({2, 3})("AB123")
Результат {"AB", "123"}
Splitter.SplitTextByPositions(positions as list, optional startAtEnd as nullable logical) as function - Повертає функцію, яка розбиває текст на список тексту в кожній вказаній позиції.
Приклад
Splitter.SplitTextByPositions({0, 3, 4})("ABC|12345")
Результат {"ABC", "|", "12345"}
Splitter.SplitTextByRanges(ranges as list, optional startAtEnd as nullable logical) as function - Повертає функцію, яка розбиває текст на список тексту відповідно до вказаних зсувів і довжини. Нульова довжина вказує на те, що всі вхідні дані, що залишилися, повинні бути включені.
Приклад Splitter.SplitTextByRanges({{0, 4}, {2, 10}})("codelimiter")
Результат {"code", "delimiter"}
Splitter.SplitTextByRepeatedLengths(length as number, optional startAtEnd as nullable logical) as function
Повертає функцію, яка багаторазово розбиває текст на список тексту після вказаної довжини.

Функції порівняння

Comparer.Equals(comparer as function, x as any, y as any) as logical - Повертає логічне значення на основі перевірки рівності двох даних значень x і y, використовуючи наданий компаратор.

comparer — описує логіку порівняння.

Порівняльник - це функція, яка приймає два аргументи та повертає -1, 0 або 1 залежно від того, чи є перше значення менше, дорівнює або більше другого.

Comparer.Equals(Comparer.FromCulture("en-US"), "1", "A") / false

Comparer.FromCulture(culture as text, optional ignoreCase as nullable logical) as function - повертає функцію порівняння, яка використовує культуру та чутлива до регістру

Comparer.FromCulture("en-US")("a", "A")/-1

Comparer.Ordinal(x as any, y as any) as number - Повертає функцію порівняння, яка використовує порядкові правила для порівняння наданих значень x і y .

Comparer.Equals(Comparer.Ordinal, "encyclopædia", "encyclopaedia")/false

Comparer.OrdinalIgnoreCase(x as any, y as any) as number

Повертає функцію порівняння без урахування регістру, яка використовує порядкові правила для порівняння значень x і y . Функція порівняння приймає два аргументи та повертає -1, 0 або 1 залежно від того, чи перше значення менше, дорівнює або більше другого.

Comparer.OrdinalIgnoreCase("Abc", "abc")/0

Функції - вирази

Expression.Constant(value as any) as text - Повертає представлення постійного значення у вихідному коді M.

Expression.Constant(123)/ "123"

Expression.Constant(#date(2035, 01, 02))/ "#date(2035, 1, 2)“

Expression.Constant("abc")/ """abc""“

Expression.Evaluate(document as text, optional environment as nullable record) as any - Повертає результат обчислення документа M-виразу з доступними ідентифікаторами на які можна посилатися, визначені середовищем.

Expression.Evaluate("1 + 1")/2

Expression.Evaluate("List.Sum({1, 2, 3})", [List.Sum = List.Sum])/6

Expression.Evaluate(Expression.Constant("""abc") & " & " & Expression.Identifier("x"), [x = "def"""])/ """abcdef""“

Expression.Identifier(name as text) as text Повертає представлення вихідного коду M імені ідентифікатора.

Expression.Identifier("MyIdentifier")/ "MyIdentifier“

Expression.Identifier("My Identifier") / "#""My Identifier""“

Функції - значення

Function.Invoke(function as function, args as list) as any - Викликає вказану функцію за допомогою вказаного та повертає результат.
Function.Invoke(Record.FieldNames, {[A = 1, B = 2]})/ {"A", "B"}

Function.InvokeAfter(function as function, delay as duration) as any - Повертає результат виклику функції після тривалості затримки пройшов.

Function.InvokeWithErrorContext(function as function, context as text) as any - функція призначена лише для внутрішнього використання.

Function.IsDataSource(function as function) as logical - повертає скалярну функцію типу scalarFunctionType, яка викликає vectorFunction з одним рядком аргументів і повертає єдиний вихід.

Лінійні функції

Lines.FromBinary(binary as binary, optional quoteStyle as any, optional includeLineSeparators as nullable logical, optional encoding as nullable number) as list - Перетворює список двійкових значень, розділених на рядків.
binary: двійкове значення для перетворення в список.
quoteStyle: визначає, як обробляються розриви рядків. Значення quoteStyle може бути null. Значенням за замовченням є QuoteStyle.None.
includeLineSeparators : вказує, чи включати символи розриву рядка. Значення includeLineSeparators може бути нульовим. Значення за замовчуванням: false. Кодування : визначає текстове кодування двійкового значення. Значення кодування може бути нульовим. Значення за замовчуванням — 65001 (UTF-8).

Lines.FromText(text as text, optional quoteStyle as any, optional includeLineSeparators as nullable logical) as list
Перетворює текстове значення на список текстових значень, розділених на рядки.
Параметри функції:
text : текстове значення, яке потрібно перетворити на список текстових значень.
quoteStyle : визначає, як обробляються розриви рядків. Значення quoteStyle може бути null.
Стандартним значенням є QuoteStyle.None. includeLineSeparators : вказує, чи включати символи розриву рядка текст.
Значення includeLineSeparators може бути нульовим. Значення за замовчуванням: false .

Функції для роботи із списками

List.Accumulate(list as list, seed as any, accumulator as function) as any - накопичує підсумкове значення з елементів у список. Дозволяє встановити необов'язковий початковий параметр seed.

List.Accumulate({1, 2, 3, 4, 5}, 0, (state, current) => state + current)/35

List.AllTrue(list as list) as logical Повертає логічне значення True, якщо всі вирази у списку дорівнюють True

Приклади

List.AllTrue({true, true, 2 > 0})/true

List.AllTrue({true, false, 2 < 0})/false

List.Alternate(list as list, count as number, optional repeatInterval as nullable number, optional offset as nullable number) as list - Повертає список, що складається з усіх непарних елементів зсуву в списку. Параметр count : визначає кількість значень, які пропускаються щоразу. offset : Параметр зміщення опції, щоб почати пропускати початкові значення зсув.

List.Alternate({1..10}, 1)/ {2, 3, 4, 5, 6, 7, 8, 9, 10}

List.Alternate({1..10}, 1, 1/ {2, 4, 6, 8, 10}

List.Alternate({1..10}, 1, 1, 1)/ {1, 3, 5, 7, 9}

List.AnyTrue(list as list) as logical – Повертає логічне значення True, якщо хочаб 1 вираз у списку дорівнюють True

List.AnyTrue({true, false, 2>0})/true

List.AnyTrue({2 = 0, false, 2 < 0})/ false

List.Average(list as list, optional precision as nullable number) as any – повертає середнє значення

List.Average({3, 4, 6})/4.3

List.Average({#date(2011, 1, 1), #date(2011, 1, 2), #date(2011, 1, 3)})/ #date(2011, 1, 2)

Функції дати і часу

DateTime.Date(dateTime as any) as nullable date - Повертає компонент дати параметра dateTime, якщо параметр є датою, datetime або значення datetimezone або null, якщо параметр має значення null.

DateTime.Date(#datetime(2010, 12, 31, 11, 56, 02))

#date(2010, 12, 31)

DateTime.FromFileTime(fileTime as nullable number) as nullable datetime - Створює значення datetime із значення fileTime і перетворює його на місцевий часовий пояс. Час файлу — це значення часу файлу Windows, яке представляє число 100 наносекунд інтервали, що минули з 12:00 опівночі 1 січня 1601 року нашої ери (н. е.) Скоординовано Всесвітній час (UTC).

DateTime.FromFileTime(129876402529842245)/ #datetime(2012, 7, 24, 14, 50, 52.9842245)

DateTime.FromText(text as nullable text, optional options as any) as nullable datetime Створює значення дати й часу з текстового представлення text . Необов'язковий запис параметр, options , може бути наданий для визначення додаткових властивостей. Запис може містять такі поля:

Формат : текстове значення, що вказує формат, який слід використовувати.

Культура: якщо формат не нульовий, культура керує деякими специфікаторами формату. для наприклад, в "en-US" "MMM" - це "Jan", "Feb", "Mar", ... , тоді як в "ru-RU" "MMM" це "янв", "фев", "мар", ... Коли Format має значення null, культура керує значенням за замовчуванням формат для використання. Якщо культура нульова або опущена, використовується культура.поточний.

Для підтримки застарілих робочих процесів параметри також можуть мати текстове значення. Це те саме поведінка, ніби options = [Format = null, Culture = options] .

DateTime.FromText("2010-12-31T01:30:25")/ #datetime(2010, 12, 31, 1, 30, 25)

DateTime.FromText("30 Dez 2010 02:04:50.369730", [Format="dd MMM yyyy HH:mm:ss.ffffff", Culture="de-DE"])/ #datetime(2010, 12, 30, 2, 4, 50.36973)