ЧЕБЫШЕВСКИЙ СБОРНИК
Том 16 Выпуск 3 (2015)

ГНЕЗДОВЫЕ МАССИВЫ И РЕКУРСИЯ

А. Р. Есаян, Н. М. Добровольский (Тула)

Аннотация

Решение задач с данными, представленными вложенными или, по-другому, гнездовыми массивами, является непростым делом из-за достаточно непредсказуемой их структуры. И здесь во многих случаях спасительной оказывается рекурсия. Ее использование позволяет линейно по одной и той же схеме осуществлять пробежку по всем элементам каждого уровня любого гнездового массива вне зависимости от его структуры и глубины вложенности. Гнездовой массив можно интерпретировать деревом, корнем которого является сам массив, от него идут дуги к массивам-элементам и т. д. Листьями подобного дерева являются скаляры или строки — конечные элементы, не имеющие ссылок на последующие массивы.

В статье для решения ряда задач общего характера с гнездовыми массивами предлагаются соответствующие рекурсивные программы-функции. Вот примеры таких задач: подсчитать общее количество листьев массива; сформировать массив из транспонированных на всех уровнях вложенности элементов исходного массива; выяснить, является ли данный объект (скаляр, строка, простой массив, гнездовой массив) элементом данного массива на каком-либо уровне вложенности; подсчитать количество вхождений объекта в массив на всех уровнях вложенности; собрать все листья массива в вектор, заместить листья данного массива элементами какого-либо вектора и т. п.

Во всех случаях рекурсивная триада такова: параметр рекурсии — гнездовой массив; декомпозиция — переходы на всех уровнях вложенности от массивов к их элементам и так до листьев; рекурсивная база, то есть тривиальные случаи в рекурсии — листья массивов [1]. Предлагаемые лаконичные рекурсивные программы-функции решения перечисленных и некоторых других задач реализованы на простом и интуитивно понятном языке программирования системы инженерных и научных вычислений *PTC Mathcad Prime* (версия 3.1) [2,3].

Отметим, что в этой системе гнездовые массивы — это вложенные друг в друга матрицы.

Ключевые слова: гнездовой массив, матрица, дерево, рекурсия, рекурсивная функция, декомпозиция, *PTC Mathcad*.

Библиография: 3 названия.

NESTED ARRAYS AND RECURSION

A. R. Esayan, N. M. Dobrovolsky (Tula)

Abstract

Problem solving with data presented nested arrays, is difficult because they of the rather is unpredictable in their structure. And here, in many cases helps recursion. Its use allows linearly according to the same scheme to implement a run on all the elements of each nesting level of any of the array, regardless of its structure and the depth of nesting. Nested array can be interpreted by a tree, whose root is the array itself, from its go arc to the array elements, etc. The leaves of this tree are scalars or strings — finite elements that are not referenced in the following arrays.

In an article for the solution of several problems of a General nature with nested arrays is offered appropriate recursive program-functions. Examples of such tasks: calculate the total number of leaves of the array; to form an array of transposed elements of the original array at all levels of nesting; determine whether a given object (scalar, string, a simple array, nested array) of a element of this array to any level of nesting; count the number of occurrences of an object in the array at all levels of nesting; collect all the leaves of the array into the vector, replacing the leaves a given array on components the vector, etc.

In all cases, recursive triad is as follows: the parameter of recursion — nest array; decomposition — transitions at all levels of nesting of arrays to their elements, and so on until the leaves; recursive base, i.e. the trivial cases in recursion — lists of arrays [1]. Offer laconic recursive programs-features of the solution are listed and some other tasks are implemented on a simple and intuitive programming language system engineering and scientific computing *PTC Mathcad Prime* (version 3.1) [2,3].

Note that in this system all nesting arrays are nested matrix.

Keywords: nested array, matrix, tree, recursion, recursive function, decomposition, *PTC Mathcad*.

Bibliography: 3 titles.

1. Введение

В математике рекурсия используется в двух аспектах: прикладном — как специальный способ нахождения решений определенных задач, и сугубо теоретическом — для формализации интуитивного понятия алгоритма. В информатике рекурсия применяется как практически ориентированное знание с изящным и эффективным инструментарием для реальных вычислений. Именно о прикладном аспекте рекурсии, связанном с гнездовыми (вложенными) массивами и пойдет речь ниже.

Решение задач с данными, представленными гнездовыми массивами, часто является непростым делом. Происходит это из-за достаточно непредсказуемой структуры таких массивов. И здесь во многих случаях спасительной оказывается рекурсия. Ее использование легко позволяет осуществлять пробежку по всем

элементам каждого уровня любого гнездового массива, совершая ту или иную обработку данных. В статье для решения ряда задач общего характера с гнездовыми массивами предлагаются соответствующие рекурсивные программы-функции. Речь идет о таких задачах:

- *для произвольных гнездовых массивов*: найти высоту; подсчитать общее количество листьев; проверить, все ли листья имеют один и тот же тип; сформировать массив из транспонированных на всех уровнях вложенности элементов; выяснить, является ли данный объект элементом массива на каком-либо уровне вложенности; подсчитать количество вхождений объекта в массив на всех уровнях вложенности; собрать все листья массива в вектор; заместить листья массива элементами вектора;
- *для гнездовых числовых массивов*: вычислить сумму и произведение листьев; найти максимальное и минимальное из значений листьев;
- *для двух произвольных гнездовых массивов* проверить одинаковость их структуры;
- *для двух гнездовых числовых массивов* найти произведение массивов, то есть массив с листьями, равными произведениям соответствующих листьев исходных массивов.

Предложенные лаконичные пользовательские рекурсивные функции решения указанных задач реализованы на простом и интуитивно понятном языке программирования системы инженерных и научных вычислений *PTC Mathcad Prime* (версия 3.1).

2. Гнездовые массивы и их высоты

Гнездовые или, по-другому, вложенные массивы — это массивы, элементами которых могут быть скаляры (числа), строки или снова массивы, причем на любой глубине вложенности. В *PTC Mathcad* массивы — это векторы или матрицы, причем векторы являются частным случаем матриц с одним столбцом. Поэтому гнездовые массивы — это, фактически, вложенные матрицы. Вот примеры таких массивов:

$$w1 := \begin{bmatrix} 1 & -3 & 7 \\ 7 & 6 & [7\ 51] \end{bmatrix} \qquad w2 := \begin{bmatrix} [7\ 9] & 3 & & & & & 7 & & \\ & & [& [& [7\ 2 & & & 3 &]]] \\ & & | & | & | & [& [& 9 &]]]] \\ & 4 & 7 & [2 & 2 & 4\ 5 & [77 & [& 1 &]]]]] \\ & & | & | & [& [& [[1\ 2\ 3]]]]] \\ & & [& [4 & & & 7 & &]]] \end{bmatrix}$$

Фрагмент 1

Высота гнездового массива

a) $hei1(ma) :=$
```
h ← 0
if IsArray(ma)
  for b ∈ ma
    h ← max(h, 1 + hei1(b))
```

b) $hei2(ma) :=$
```
h ← 0
if IsArray(ma)
  1 + max(⟨hei2(ma)⟩)
```

c) $hei3(ma) := \text{if}\left(\text{IsArray}(ma), 1 + \max\left(\overrightarrow{hei3(ma)}\right), 0\right)$

d) $mahei(ma, h) :=$
```
if IsArray(ma)
  for i ∈ 0..rows(ma) − 1
    for j ∈ 0..cols(ma) − 1
      b ← ma_{i,j}
      ma_{i,j} ← if(IsArray(b), mahei(b, h + 1), h)
else
  ma ← h − 1
ma
```

$hei4(ma) := mahei(ma, 1)$ - это головная программа

$$a := \begin{bmatrix} 1 & 3 & 7 \\ 4 & \text{“x”} & 9 \end{bmatrix} \quad b := \begin{bmatrix} 1 & 3 & 7 \\ 4 & 6 & [\text{“yy”} \;\; 5] \end{bmatrix} \quad c := \begin{bmatrix} [7 \;\; 9] & 3 & 7 \\ 4 & 6 & \begin{bmatrix} 2 & \begin{bmatrix} 1 & \text{“2”} & 3 \\ 4 & 5 & [77 \;\; 2] \end{bmatrix} \\ \begin{bmatrix} 4 & [7] \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$hei1(5) = 0 \quad hei1(a) = 1 \quad hei1(b) = 2 \quad hei1(c) = 5$

$hei2(5) = 0 \quad hei2(a) = 1 \quad hei2(b) = 2 \quad hei2(c) = 5$

$hei3(5) = 0 \quad hei3(a) = 1 \quad hei3(b) = 2 \quad hei3(c) = 5$

$hei4(5) = 0 \quad hei4(a) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad hei4(b) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & [2 \;\; 2] \end{bmatrix}$

$$hei4(c) = \begin{bmatrix} [2 \;\; 2] & 1 & 1 \\ 1 & 1 & \begin{bmatrix} 2 & \begin{bmatrix} 3 & \begin{bmatrix} 4 & 4 & 4 \\ 4 & 4 & [5 \;\; 5] \end{bmatrix} \\ \begin{bmatrix} 3 & [4] \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Гнездовой массив можно интерпретировать деревом, корнем которого является сам массив, от него идут дуги к массивам-элементам и т. д. Листья подобного дерева — это скаляры или строки. Они являются конечными элементами и не имеют ссылок на последующие массивы. Высотой гнездового массива назовем высоту соответствующего ему дерева, то есть максимальное из количеств дуг на путях от корня до листьев. Фактически высота массива есть глубина его вложенности. Определить высоту раскрытого гнездового массива можно подсчетом в нем количества открывающих (или закрывающих) квадратных скобок.

Решение многих задач, связанных с гнездовыми массивами, требует знания их высот. Три простых рекурсивных функции $hei1$, $hei2$ и $hei3$ для вычисления этой характеристики массивов показаны на фрагменте 1. Там же приведена программа из пары функций $mahei$ и $hei4$, которая по исходному массиву выводит новый массив такой же структуры с листьями, равными глубинам их вложенности. Кратко остановимся на описании перечисленных функций.

Функция $hei1$. Аргументом $hei1$ может быть скаляр, строка или массив ma. Декомпозиция в рекурсии реализуется переходами к подмассивам нижнего уровня. Тривиальные случаи задаются листьями. Иными словами, если ma — скаляр или строка, то возвращается высота $h = 0$, в противном случае для каждого $b \in ma$ происходит рекурсивное обращение $hei1(b)$ и вычисляется h, равное максимальному из текущего значения h и $1 + hei1(b)$. Этим и обеспечивается подсчет высоты h исходного гнездового массива.

Функция $hei2$. По этой функции, в отличие от $hei1$, для пробежки по элементам массива на каждом уровне вложенности вместо цикла используется оператор векторизации.

Функция $hei3$. Данная функция фактически дублирует $hei2$, но здесь оформление ветвей реализуется с помощью функции if.

Функции $mahei$ и $hei4$. Здесь $hei4$ — головная функция, $mahei$ — рекурсивная функция. Из $hei4$ в $mahei$ передается исходный гнездовой массив ma и начальное значение высоты $h = 1$. По $mahei$ проводятся все основные вычисления и результат формируется на месте матрицы ma. Делается это так. Если ma не есть массив, то есть скаляр или строка, то возвращается $h - 1$ (0) и вычисления прекращаются. В противном случае реализуется пробежка по элементам ma. Если очередной элемент b есть лист, то ему присваивается значение h, иначе погружаемся в следующий рекурсивный вызов со значением $h = h + 1$ и т. д. Все это и обеспечивает требуемое изменение листьев ma.

3. Некоторые задачи и функции для их решения

Для перечисленных ниже простых задач, связанных с гнездовыми массивами, приведены варианты рекурсивных программ-функций для нахождения их решений и дано краткое описание этих функций. Формулировки задач таковы:

А. Найти общее количество листьев гнездового массива, то есть количество

его конечных скалярных и строковых элементов всех уровней вложенности;

B. Найти сумму листьев гнездового числового массива;

C. Найти произведение листьев гнездового числового массива;

D. Найти максимальное значение из листьев гнездового числового массива;

E. Найти минимальное значение из листьев гнездового числового массива;

F. Выяснить, имеются ли в гнездовом массиве ma на каком-либо уровне вложенности элементы, совпадающие с объектом x (x, ma — не обязательно числовые; x — скаляр, строка, простой или гнездовой массив). При $x \in ma$ функция должна возвращать 1 (да), при $x \notin ma$ — 0 (нет);

G. Подсчитать, сколько элементов гнездового массива ma на всех уровнях вложенности совпадают с объектом x (x, ma — не обязательно числовые; x — скаляр, строка, простой или гнездовой массив);

H. Транспонировать гнездовой массив и все его подмассивы на всех уровнях вложенности;

I. Выяснить, имеют ли одинаковую структуру два гнездовых массива;

J. Проверить, все ли листья гнездового массива имеют один и тот же тип ($IsScalar, IsString, IsNaN, \dots$). Тип листьев может быть пользовательским;

K. По двум гнездовым числовым массивам одинаковой структуры построить массив такой же структуры с листьями, равными произведения соответствующих листьев исходных массивов;

L. По двум гнездовым числовым массивам одинаковой структуры построить массив такой же структуры с листьями, равными наибольшим общим делителям соответствующих листьев исходных массивов.

Перечисленные задачи можно решать рекурсивными функциями, представленными на фрагментах 2-6. Для некоторых задач приводятся по два варианта решения. Кратко опишем выполнение соответствующих функций.

A. Задача решается функцией $nu(ma)$, где ma — произвольный гнездовой массив, листьями которого могут быть как числа, так и строки. Устроена функция nu так. Подсчитывается общее количество элементов su текущего массива ma ($su \leftarrow rows(ma) \cdot cols(ma)$). Далее, в цикле совершается пробежка по всем элементам $b \in ma$. Если окажется, что очередной элемент $b \in ma$ не массив, то величина su не меняется. В противном случае реализуется рекурсивный вызов $nu(b)$ и так далее. Поскольку на выходе из любого рекурсивного вызова к su добавляется количество конечных элементов обследуемого массива, то в конечном счете su становится равным общему количеству листьев исходного массива. Наращивание su происходит по формуле $su \leftarrow su + nu(b) - 1$ потому, что добавляя к su количество листьев очередного массива b, необходимо вычитать 1, ибо до этого b учитывался как лист.

Замечание. Количество листьев массива можно сразу учитывать точно, минуя последующие поправки. Для этого первую строку nu следует изменить на $su \leftarrow 0$, а тело цикла — на $su \leftarrow su + if(IsArray(b), nu(b), 1)$. Эти изменения и осуществлены в функции num.

B. Задача решается функциями $sum1(ma)$ и $sum2(ma)$, где ma — произвольный гнездовой числовой массив. Здесь функция $sum1$ построена аналогично функции num задачи A. Функция $sum2$ — это другая форма записи функции $sum1$, в которой цикл по элементам массива заменен двойной суммой, что позволило избавиться от присваивания $su \leftarrow 0$.

C. Задача решается функциями $mul1(ma)$ и $mul2(ma)$, где ma — произвольный гнездовой числовой массив. Здесь функция $mul1$ построена аналогично функции $sum1$ задачи B. Для получения $mul1$ в первой строке $sum1$ потребовалось заменить оператор $su \leftarrow 0$ на $pr \leftarrow 1$, а в теле цикла оператор "+" на ".". Функция $mul2$ — это другая форма записи функции $mul1$, в которой двойной цикл по элементам массива заменен двойным произведением, что позволило избавиться от присваивания $pr \leftarrow 1$.

D. Задача решается функцией $mmax(ma)$, где ma — гнездовой числовой массив. Устроена функция $mmax$ так. На любой глубине рекурсивных вызовов проверяется, является ли очередной элемент b массивом. Если да, то реализуется следующий рекурсивный вызов, иначе сразу находится максимальное из числа b и ранее найденных значений. Заметим, что в тексте $mmax$ встроенная функция max применяется лишь к паре чисел. В общем случае функция max может работать только с числами и простыми числовыми массивами, а сформированная функция $mmax$ является ее обобщением на гнездовые массивы. Используя обе функции можно находить максимальный элемент из последовательности чисел, простых числовых массивов и гнездовых числовых массивов. Если, скажем, a и b — простые числовые массивы, а c и d — гнездовые числовые массивы, то максимальное значение из 7 и элементов этих массивов можно искать, например, так: $max(7, a, b, mmax(c), mmax(d))$.

E. Из функции $mmax$ легко построить новую рекурсивную функцию $mmin$ для нахождения минимального из листьев гнездового числового массива. Для этого кроме изменения названия функции в ее теле в первой строке у символа "∞" следует убрать знак "$-$", а в последней строке функцию max заменить функцией min.

F. Задача решается функцией $in(x, ma)$, где x — объект, ma — гнездовой массив. Выполняется in так. Сначала проверяется, совпадает ли x с ma и результат (1 — да, 0 — нет) заносится в переменную su. Если $su = 1$, то вычисления надо бы прекращать, но делается это чуть позже. Далее, реализуется пробежка по элементам $b \in ma$. Если $x = b$ или вхождение было найдено в других рекурсивных вызовах ($su = 1$), то текущий вызов прерывается по "$return$ 1". В противном случае формируется переменная su: если b — скаляр или строка,

то $su = 0$, если b — массив, то su равно результату рекурсивного обращения $in(x, b)$. Таким образом, в выводимой переменной su окажется 1, если хотя бы раз выполнился оператор "$return$ 1", и 0 — в противном случае.

G. Задача решается функцией $numx(x, ma)$, где x — объект, ma — гнездовой массив. Выполняется $numx$ так. Сначала проверяется, совпадает ли x с ma и результат (1 — да, 0 — нет) заносится в переменную su. Далее, начинает осуществляться пробежка по элементам $b \in ma$. При совпадениях $x = b$ величина su увеличивается на 1. В противном случае формирование su продолжается так: если b — скаляр или строка, то su не меняется, если b — массив, то к su добавляется результат рекурсивного обращения $numx(x, b)$. Таким образом, после вычислений в выводимой переменной su во всех случаях оказывается количество вхождений x в ma.

H. Задача решается функцией $tra(ma)$, где ma — гнездовой массив. Заметим, что встроенная операция транспонирования "T" работает и с простыми массивами, и с гнездовыми массивами, но реализуется только на верхнем уровне вложенности. То есть, при выполнении обычного транспонирования строки и столбцы ma меняются местами, но сами элементы ma, даже если это массивы, никаким образом не преобразуются. Функция tra транспонирует массивы ma на всех уровнях вложенности. Устроена она так. Сначала транспонируется исходный массив ma. Далее, в цикле совершается пробежка по элементам b из ma и для каждого из них, являющегося массивом, реализуется рекурсивное обращение $tra(b)$. Этим и обеспечивается транспонирование массивов на всех уровнях вложенности ma.

I. Задача решается функцией $stru(x, y)$, где x, y — массивы. Заметим, что два простых массива имеют одинаковую структуру, если у них совпадают количества строк и количества столбцов. Будем считать, что два гнездовых массива имеют одинаковую структуру, если в них, во-первых, количества строк и столбцов совпадают и, во-вторых, на всех уровнях вложенности соответствующие друг другу элементы одновременно массивы или не массивы, а в соответствующих друг другу массивах количества строк и количества столбцов совпадают. Это определение и положено в основу написания функция $stru$. Оператор "$return$ 0" используется при завершении вычислений по текущему рекурсивному вызову в том случае, если уже обнаружено несовпадение структур массивов x и y.

J. Задача решается функцией $isg(ma, g)$, где ma — массив, g — встроенная или пользовательская логическая функция $(IsScalar, IsString, \dots)$. Работает isg следующим образом. На очередном рекурсивном вызове, если ma не массив, то в переменную pri заносится результат применения g к ma ($pri \leftarrow g(ma)$). Если же ma — массив, то реализуется пробежка по элементам $b \in ma$, при которой прежнее значение переменной pri умножается на результат рекурсивного вызова $isg(b, g)$. Все это обеспечивает окончательное значение $pri = 1$, если

условию g удовлетворяют все листья, и $pri = 0$, если условию g не удовлетворяет хотя бы один лист. Обратите внимание, что функция isg работает не только на массивах. Например, $isg(7, IsScalar) = 1$, $isg("7", IsString) = 1, \ldots$. Еще раз подчеркнем что g может быть не только встроенной, но и пользовательской функцией. Если, скажем, требуется проверить, будут ли листья гнездового числового массива числами, кратными 5, то сделать это можно, например, по "$isg(ma, div5) =$", где $div5(x) := if(mod(x, 5), 1, 0)$.

K. Задача решается функциями $mul(x, y)$ и $mulrec(x, y)$, где x, y — гнездовые массивы. Вычисления завершаются в головной программе mul в следующих случаях: x и y скаляры (возвращается их произведение), x и y имеют разную структуру или в них имеются не скалярные листья (возвращается соответствующее сообщение). В противном случае происходит обращение к рекурсивной функции $mulrec(x, y)$. Устроена она так. В двойном цикле for обрабатывается каждая пара соответствующих элементов a и b из x и y. Если это листья, то находится их произведение, если массивы, то реализуется рекурсивное обращение $mulrec(a, b)$. Результат формируется в матрице B такой же структуры, что и аргументы x и y.

4. Листья гнездовых массивов и векторы

Остановимся еще на одном вопросе, связанном с гнездовыми массивами. Решения некоторых задач не используют структуру гнездового массива, а опираются лишь на данные, представленные его листьями. В таких случаях полезными могут быть средства, собирающие все листья массива ma в один простой массив, например, в вектор ve. Обрабатывать ve уже можно не рекурсивными программами, что обычно проще. Далее, если, решая задачу, из вектора ve мы получили некоторый новый вектор we, то могут потребоваться средства, с помощью которых можно заменить листья исходного массива ma последовательными компонентами we. Эти рассуждения приводят нас к постановке таких задач:

A. Написать рекурсивную функцию, собирающую все листья гнездового массива ma в простой вектор ve;

B. Написать рекурсивную функцию, по которой реализуется замещение последовательных листьев гнездового массива ma последовательными элементами вектора we. При этом we — не обязательно простой вектор (его элементами могут быть числа, строки и массивы) и количество листьев в ma не обязательно равно количеству компонентов we.

Решение задачи A. На фрагменте 3 приведены две функции $mavec1$ и $mavec2$, решающие данную задачу. Вторая из них является компактной записью первой. Поэтому кратко остановимся лишь на описании выполнения $mavec1(ma)$.

Фрагмент 2 (1,2,3,4)

Примеры. Рекурсивные программы с гнездовыми массивами

Массивы для расчетов:

$$a := \begin{bmatrix} 1 & 3 & 7 \\ 4 & 6 & 9 \end{bmatrix} \qquad b := \begin{bmatrix} 1 & 3 & 7 \\ 7 & 6 & [7\ 5] \end{bmatrix} \qquad b1 := \begin{bmatrix} 1 & 3 & 7 \\ 7 & 6 & [7\ 5] \\ & & [2\ 1] \end{bmatrix}$$

$$c := \begin{bmatrix} [7\ 9] & 3 & & 7 & \\ & & [& [& 7 & 2 & 30 &]]] \\ 4 & 7 & 2 & [& 4 & 5 & [7\ 9] &]]] \\ & & [& [7\ 9] & & 7 & &]]] \end{bmatrix} \qquad d := \begin{bmatrix} [7\ 9] & 3 & & 7 & \\ & & [& [& 7 & 2 & 30 &]]] \\ 4 & 7 & 2 & [& 4 & 5 & [7\ \text{“9”}] &]]] \\ & & [& [7\ 9] & & 7 & &]]] \end{bmatrix}$$

A) Подсчет общего количества листьев гнездового массива (конечных скалярных и строковых элементов)

A1) $nu1(ma) := \left| \begin{array}{l} su \leftarrow \text{rows}(ma) \cdot \text{cols}(ma) \\ \text{for } b \in ma \\ \quad su \leftarrow su + \text{if}(\text{IsArray}(b), nu1(b) - 1, 0) \end{array} \right.$

$nu1(a) = 6$
$nu1(b) = 7$
$nu1(c) = 18$
$nu1(d) = 18$

A2) $nu2(ma) := \left| \begin{array}{l} su \leftarrow 0 \\ \text{for } b \in ma \\ \quad su \leftarrow su + \text{if}(\text{IsArray}(b), nu2(b), 1) \end{array} \right.$

$nu2(a) = 6$
$nu2(b) = 7$
$nu2(c) = 18$
$nu2(d) = 18$

A3) $nu3(ma) := \sum_{i=0}^{\text{rows}(ma)-1} \sum_{j=0}^{\text{cols}(ma)-1} \text{if}\left(\text{IsArray}\left(ma_{i,j}\right), nu3\left(ma_{i,j}\right), 1\right)$ $\quad nu3(a) = 6$

B) Вычисление суммы конечных элементов всех уровней вложенности гнездового числового массива

B1) $sum1(ma) := \left| \begin{array}{l} su \leftarrow 0 \\ \text{for } b \in ma \\ \quad su \leftarrow su + \text{if}(\text{IsArray}(b), sum1(b), b) \end{array} \right.$

$sum1(a) = 30$
$sum1(b) = 36$
$sum1(c) = 128$

B2) $sum2(ma) := \sum_{i=0}^{\text{rows}(ma)-1} \sum_{j=0}^{\text{cols}(ma)-1} \text{if}\left(\text{IsArray}\left(ma_{i,j}\right), sum2\left(ma_{i,j}\right), ma_{i,j}\right)$

$sum2(a) = 30 \qquad sum2(b) = 36 \qquad sum2(c) = 128$

C) Вычисление произведения конечных элементов всех уровней вложенности гнездового числового массива

C1) $mul1(ma) := \left| \begin{array}{l} pr \leftarrow 1 \\ \text{for } b \in ma \\ \quad pr \leftarrow pr \cdot \text{if}(\text{IsArray}(b), mul1(b), b) \end{array} \right.$

Фрагмент 2 (1,2,3,4)

C2) $mul2(ma) := \prod_{i=0}^{rows(ma)-1} \left(\prod_{j=0}^{cols(ma)-1} \text{if}\left(\text{IsArray}\left(ma_{i,j}\right), mul2\left(ma_{i,j}\right), ma_{i,j}\right) \right)$

$mul1(a)=4536$ $mul1(b)=30870$ $mul1(c)=34580899987200$
$mul2(a)=4536$ $mul2(b)=30870$ $mul2(c)=34580899987200$

D) Нахождение максимального значения из листьев гнездового числового массива

```
mmax(ma) := mm ← -∞
            for b ∈ ma
              mm ← max(if(IsArray(b), mmax(b), b), mm)
```

$mmax(a)=9$
$mmax(b)=7$
$mmax(c)=30$
$mmax(-c)=-2$

$\max(a)=9$ $\max(5,a,mmax(b),mmax(c))=30$

E) Нахождение минимального значения из листьев гнездового числового массива

```
mmin(ma) := mm ← ∞
            for b ∈ ma
              mm ← min(if(IsArray(b), mmin(b), b), mm)
```

$mmin(a)=1$
$mmin(b)=1$
$mmin(c)=2$
$mmin(-c)=-30$

F) Проверка, что объект *x* является элементом гнездового массива *ma* на какой-либо глубине его вложенности

```
in(x, ma) := su ← if(x = ma, 1, 0)
             for b ∈ ma
               if (su = 1) ∨ (b = x)
                 return 1
               else
                 su ← if(IsArray(b), in(x, b), 0)
```

$in(6,a)=1$ $in(5,b)=1$
$in(78,c)=0$ $in(c,c)=1$
$in([7\ \ 5],b)=1$
$in([7\ \ 5],b1)=0$
$in([7\ \ 9],c)=1$
$in(\text{“5”},b)=0$

$in\left(\begin{bmatrix}4 & 5\\ 6 & 7\end{bmatrix},c\right)=0$

$in\left(\begin{bmatrix}2 & \begin{bmatrix}7 & 2 & 30\end{bmatrix}\\ \begin{bmatrix}7 & 9\end{bmatrix} & \begin{matrix}4 & 5 & \begin{bmatrix}7 & 9\end{bmatrix}\end{matrix}\end{bmatrix},c\right)=1$ $in\left(\begin{bmatrix}7 & 2 & 30\\ 4 & 5 & \begin{bmatrix}7 & 9\end{bmatrix}\end{bmatrix},c\right)=1$

$in(\text{“9”},d)=1$
$in(\text{“8”},d)=0$

G) Подсчет количества элементов гнездового массива *ma* на любой глубине его вложеннности, совпадающих с объектом *x*

```
numx(x, ma) := su ← if(x = ma, 1, 0)
               for b ∈ ma
                 if b = x
                   su ← su + 1
                 else
                   su ← su + if(IsArray(b), numx(x, b), 0)
```

$numx(7,b)=3$
$numx(7,c)=7$
$numx([7\ \ 9],c)=3$
$numx([7\ \ 9],d)=2$

Фрагмент 2 (1,2,3,4)

$$numx\left(\begin{bmatrix} \begin{bmatrix} \\ 2 \\ \end{bmatrix} & \begin{bmatrix} \\ 2 \\ \end{bmatrix} & \begin{bmatrix} 7 & 2 & 30 \\ 4 & 5 & \begin{bmatrix} 7 & \text{“9”} \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \\ \begin{bmatrix} 7 & 9 \end{bmatrix} \end{bmatrix} & & \begin{bmatrix} \\ 7 \\ \end{bmatrix} \end{bmatrix}, d\right) = 1 \qquad numx\left(\begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix}, d\right) = 0$$

H) Транспонирование гнездовых массивов и их подмассивов на всех уровнях вложенности

```
tra(ma) := ma ← ma^T
           for i ∈ 0..rows(ma)−1
             for j ∈ 0..cols(ma)−1
               b ← ma_{i,j}
               if IsArray(b)
                 ma_{i,j} ← tra(b)
           ma
```

$$tra(d) = \begin{bmatrix} \begin{bmatrix} 7 \\ 9 \end{bmatrix} & & 4 & \\ 3 & & 7 & \\ & \begin{bmatrix} & 2 & \end{bmatrix} & \\ & \begin{bmatrix} & 2 & & \begin{bmatrix} 7 \\ 9 \end{bmatrix} \end{bmatrix} & \\ 7 & \begin{bmatrix} 7 & 4 \\ 2 & 5 \end{bmatrix} & 7 & \\ & \begin{bmatrix} 30 & \begin{bmatrix} 7 \\ \text{“9”} \end{bmatrix} \end{bmatrix} & & \end{bmatrix}$$

$$tra(b) = \begin{bmatrix} 1 & 7 \\ 3 & 6 \\ & \begin{bmatrix} 7 \end{bmatrix} \\ 7 & \begin{bmatrix} 5 \end{bmatrix} \end{bmatrix} \qquad b1^{T} = \begin{bmatrix} 1 & 7 \\ 3 & 6 \\ & \begin{bmatrix} 7 & 5 \end{bmatrix} \\ 7 & \begin{bmatrix} 2 & 1 \end{bmatrix} \end{bmatrix} \qquad tra(b1) = \begin{bmatrix} 1 & 7 \\ 3 & 6 \\ & \begin{bmatrix} 7 & 2 \end{bmatrix} \\ 7 & \begin{bmatrix} 5 & 1 \end{bmatrix} \end{bmatrix}$$

I) Проверка, что два гнездовых массива имеют одинаковую структуру

```
stru(x,y) := pri ← if(rows(x)≠rows(y) ∨ (cols(x)≠cols(y)), 0, 1)
             if pri
               for i ∈ 0..rows(x)−1
                 for j ∈ 0..cols(x)−1
                   [a  b] ← [x_{i,j}  y_{i,j}]
                   if IsArray(a) ∧ IsArray(b)
                     pri ← stru(a,b)
                   else if IsArray(a) ∨ IsArray(b)
                     return 0
             pri
```

$$stru(b,b1) = 0$$

$$stru(c,d) = 1$$

$$stru\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, 2\right) = 0 \qquad stru(b,c) = 0 \qquad stru(c,c+2) = 1$$

$$stru\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = 1 \qquad stru\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ \text{“1”} \end{bmatrix}\right) = 1 \qquad stru\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}\right) = 0$$

J) Проверка, что в гнездовом массиве конечные элементы (листья) имеют один и тот же тип g (IsScalar, IsString, IsNaN)

Фрагмент 2 (1,2,3,4)

```
isg(ma, g) := pri ← 1
              if IsArray(ma)
                 for b ∈ ma
                    pri ← pri · isg(b, g)
              else
                 pri ← g(ma)
```

$$x := \begin{bmatrix} \text{“1”} & \text{“3”} & \text{“7”} \\ \text{“7”} & \text{“6”} & [\text{“7”}\ \text{“5”}] \end{bmatrix} \qquad y := \begin{bmatrix} NaN & NaN & NaN \\ NaN & NaN & [NaN\ NaN] \end{bmatrix}$$

$isg(7, \text{IsScalar}) = 1$ $isg(d, \text{IsScalar}) = 0$
$isg(a, \text{IsScalar}) = 1$ $isg(c, \text{IsString}) = 0$
$isg(b, \text{IsScalar}) = 1$ $isg(d, \text{IsString}) = 0$
$isg(c, \text{IsScalar}) = 1$ $\text{IsScalar}(NaN) = 1$

$isg(x, \text{IsScalar}) = 0$ $isg(x, \text{IsString}) = 1$ $isg(x, \text{IsNaN}) = 0$
$isg(y, \text{IsScalar}) = 1$ $isg(y, \text{IsString}) = 0$ $isg(y, \text{IsNaN}) = 1$

Пользовательские типы (*IsEven* и *IsOdd*):

$IsEven(x) := \text{if}(\text{mod}(x, 2) = 0, 1, 0)$ $IsOdd(x) := \text{if}(\text{mod}(x, 2) = 1, 1, 0)$
$isg(a, IsEven) = 0$ $isg(b, IsEven) = 0$ $isg(b1, IsEven) = 0$

$A := 2 \cdot a$ $A = \begin{bmatrix} 2 & 6 & 14 \\ 8 & 12 & 18 \end{bmatrix}$ $isg(A, IsEven) = 1$

$$C := 2 \cdot c + 1 = \begin{bmatrix} [15\ \ 19] & 7 & 15 \\ 9 & 15 & \begin{bmatrix} 5 & \begin{bmatrix} 5 & [15\ \ 5\ \ 61] \\ [15\ \ 19] & [9\ \ 11\ \ [15\ \ 19]] \end{bmatrix} \\ & & 15 \end{bmatrix} \end{bmatrix}$$

$isg(C, IsOdd) = 1$

K) Нахождение "произведения" двух гнездовых числовых массивов одинаковой структуры

```
mulrec(x, y) := B ← x
                for i ∈ 0 .. rows(x) − 1
                   for j ∈ 0 .. cols(x) − 1
                      [a  b] ← [x_{i,j}  y_{i,j}]
                      B_{i,j} ← if(IsArray(a), mulrec(a, b), a · b)
                B
```

$$x := \begin{bmatrix} [1] & 3 \\ [2] & \\ 1 & [2\ \ 4] \end{bmatrix} \qquad y := \begin{bmatrix} [0] & 2 \\ [1] & \\ 5 & [3\ \ 2] \end{bmatrix}$$

```
mul(x, y) := if IsScalar(x) ∧ IsScalar(y)
                x · y
             else if ¬stru(x, y)
                "Структуры массивов разные"
             else if ¬isg(x, IsScalar) ∨ ¬isg(y, IsScalar)
                "Есть не скалярные листья"
             else
                mulrec(x, y)
```

$mul(7, 8) = 56$

$$mul(a, a) = \begin{bmatrix} 1 & 9 & 49 \\ 16 & 36 & 81 \end{bmatrix}$$

$$mul(x, y) = \begin{bmatrix} [0] & 6 \\ [2] & \\ 5 & [6\ \ 8] \end{bmatrix}$$

В ней декомпозиция проводится переходами на всех уровнях вложенности ma от матриц к их элементам и так до листьев. Обработка каждого листа — это и есть тривиальные случаи в рекурсии. А проводится эта обработка в ветви $else$ оператора if следующим образом. Если вектор ve еще не пополнялся ($ve = \infty$), то он полагается равным текущему листу b, иначе к имеющемуся ve с конца добавляется лист b ($ve = stack(ve, b)$). Декомпозиция осуществляется так. Если очередной элемент b исследуемого массива есть массив, то организуется рекурсивное обращение $r \leftarrow mavec1(b)$ и результат вычислений r или заносится в ve, если вектор ve еще не пополнялся ($ve = \infty$), или добавляется с конца к имеющемуся вектору ve ($ve = stack(ve, r)$).

Анализ контрольных примеров показывает, что листья по $mavec1(ma)$ собираются в вектор ve так. Начинается пробежка по элементам столбцов ma. Если встречается лист, то он добавляется к ve с конца. Если встречается матрица, то реализуется вход в нее и далее опять начинается пробежка по элементам ее столбцов и т. д. При необходимости можно было бы организовать сбор листьев в ve пробежкой по элементам строк каждой матрицы. Для этого вместо цикла "$for\ b \in ma \ldots$" следовало бы записать вложенный цикл:

$$for\ j \in 0..cols(ma) - 1$$
$$\quad for\ i \in 0..rows(ma) - 1 \ldots$$

и в теле цикла оперировать элементами $ma_{i,j}$.

Решение задачи B. Для решения данной задачи предлагается программа из трех функций $num(ma)$, $vecma(ma, we, k)$ и $subs(ma, we)$, представленных на фрагменте 3. По рекурсивной функции $num(ma)$ подсчитывается общее количество листьев гнездового массива ma. Основные вычисления, то есть замещение листьев в ma элементами из we, проводятся рекурсивной функцией $vecma(ma, we, k)$, где k — вспомогательный аргумент, в который из головной функции $subs$ передается 0. В дальнейшем параметр k используется в качестве сквозного индекса при выборке элементов из we. Кратко опишем выполнение функции $vecma(ma, we, k)$.

Начинается выполнение $vecma$ с проверки. Если ma не массив, то возвращается ma и на этом выполнение программы прекращается. В противном случае организуется пробежка по элементам последовательных столбцов ma от нулевого и далее, причем внутри столбцов перемещение реализуется сверху вниз. Если элементы вектора we еще не исчерпаны, то есть $k \leqslant last(we)$, то для очередного элемента $b = ma_{i,j}$ выполняется такая проверка. Если b — массив, то проводится рекурсивный вызов $ma_{i,j} \leftarrow vecma(b, we, k)$. На выходе из него мы должны подправить индекс k, увеличив его на количество произведенных в b замен листьев элементами из we. Но их было столько, сколько листьев в b. Иными словами, правка должна быть такой: $k \leftarrow k + sum(b)$. Остается рассмотреть случай, когда b — не массив. Попав на лист $b = ma_{i,j}$, мы должны сразу же заменить его текущим элементом we_k из we ($ma_{i,j} \leftarrow we_k$) и не забыть сместить индекс k ($k \leftarrow k + 1$). Обратите внимание, что если в векторе we элементов

Фрагмент 3(1-2)

Задача А. *Преобразование гнездового массива в простой вектор*

A1)
```
mavec1(ma) := ve ← ∞
              for b ∈ ma
                if IsArray(b)
                  r ← mavec1(b)
                  ve ← if(ve = ∞, r, stack(ve, r))
                else
                  ve ← if(ve = ∞, [b], stack(ve, b))
```

$$a := \begin{bmatrix} 1 & 3 & 7 \\ 4 & 6 & 9 \end{bmatrix} \qquad b := \begin{bmatrix} 1 & 3 & 7 \\ 7 & 6 & [7\ 5] \end{bmatrix} \qquad d := \begin{bmatrix} [7\ 9] & 3 & & 7 & \\ & & \begin{bmatrix} & \begin{bmatrix} 2 \end{bmatrix} & \begin{bmatrix} 7 & 2 & 30 \end{bmatrix} \end{bmatrix} \\ 4 & 7 & \begin{bmatrix} 2 & & \begin{bmatrix} 4 & 5 & [7\ 9] \end{bmatrix} \end{bmatrix} \\ & & \begin{bmatrix} [7\ 9] & 7 \end{bmatrix} \end{bmatrix}$$

$mavec1(a)^{T} = [1\ 4\ 3\ 6\ 7\ 9]$ $\quad mavec1(b)^{T} = [1\ 7\ 3\ 6\ 7\ 7\ 5]$

$mavec1(d) = [7\ 9\ 4\ 3\ 7\ 7\ 2\ 2\ 7\ 9\ 7\ 4\ 2\ 5\ 30\ 7\ 9\ 7]$

$ve := mavec1(d)$ $\quad min(ve) = 2$ $\quad \max(ve) = 30$

$\mathrm{mean}(ve) = 7.111$ - среднее значение

$\mathrm{median}(ve) = 7$ - медиана

```
summ(ma) := ve ← mavec1(ma)
            ∑ ve
```

$summ(b) = 36$ $\qquad summ(d) = 128$

A2)
```
mavec2(ma) := ve ← ∞
              for b ∈ ma
                r ← if(IsArray(b), mavec2(b), [b])
                ve ← if(ve = ∞, r, stack(ve, r))
```

$mavec2(a)^{T} = [1\ 4\ 3\ 6\ 7\ 9]$ $\quad mavec2(b)^{T} = [1\ 7\ 3\ 6\ 7\ 7\ 5]$

$mavec2(d) = [7\ 9\ 4\ 3\ 7\ 7\ 2\ 2\ 7\ 9\ 7\ 4\ 2\ 5\ 30\ 7\ 9\ 7]$

Задача В. *Замещение листьев*. Замещение последовательных листьев гнездового массива *ma* последовательными элементами вектора *ve*. (Функции:
num(ma) - подсчет количество листьев ma;
vecma(ma, we, k) - организация замещений листьев элементами ve, начиная с we_k
subs(ma,we) - головная программа для подачи в mavec ma, ve и начального k=0)

```
num(ma) := su ← 0
           for b ∈ ma
             su ← su + if(IsArray(b), num(b), 1)
```

Фрагмент 3(1-2)

```
vecma(ma, we, k) := if IsArray(ma)
                      for j ∈ 0..cols(ma) − 1
                        for i ∈ 0..rows(ma) − 1
                          if k ≤ last(we)
                            b ← ma_{i,j}
                            if IsArray(b)
                              ma_{i,j} ← vecma(b, we, k)
                              k ← k + num(b)
                            else
                              ma_{i,j} ← we_k
                              k ← k + 1
                    ma
```

$subs(ma, we) := vecma(ma, we, 0)$ - это головная программа

1) $subs(999, [77]) = 999$

$ve := [7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12]^T \qquad subs(a, ve) = \begin{bmatrix} 7 & 9 & 11 \\ 8 & 10 & 12 \end{bmatrix}$

$we := [7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13]^T \qquad subs(b, we) = \begin{bmatrix} 7 & 9 & 11 \\ 8 & 10 & [12\ \ 13] \end{bmatrix}$

$k := 1, 2..18 \qquad te_{k-1} := k \qquad te^T = [1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ \ldots]$

$$subs(d, te) = \begin{bmatrix} [1\ \ 2] & 4 & & & 6 & & \\ & & & & & & \\ & 3 & 5 & \begin{bmatrix} 7 & \begin{bmatrix} & \begin{bmatrix} 8 & \end{bmatrix} & \end{bmatrix} & \begin{bmatrix} 11 & 13 & 15 & \end{bmatrix} \\ & & & & & & \begin{bmatrix} 12 & 14 & [16\ \ 17] \end{bmatrix} \end{bmatrix} & \\ & & & \begin{bmatrix} [9\ \ 10] & & & 18 & & \end{bmatrix} & \end{bmatrix}$$

2) $ve := \begin{bmatrix} 111 \\ a \\ b \\ 777 \\ 888 \end{bmatrix} \qquad subs(d, ve) = \begin{bmatrix} \begin{bmatrix} 111 & \begin{bmatrix} 1 & 3 & 7 \\ 4 & 6 & 9 \end{bmatrix} \end{bmatrix} & 777 & & & 7 & \\ & & & & & \\ \begin{bmatrix} 1 & 3 & 7 \\ 7 & 6 & [7\ \ 5] \end{bmatrix} & & 888 & \begin{bmatrix} 2 & \begin{bmatrix} & \begin{bmatrix} 2 & \end{bmatrix} & \end{bmatrix} & \begin{bmatrix} 7 & 2 & 30 \\ 4 & 5 & [7\ \ 9] \end{bmatrix} \\ & & & \begin{bmatrix} [7\ \ 9] & & 7 & \end{bmatrix} & \end{bmatrix} \end{bmatrix}$

3) $ve := \begin{bmatrix} [10] \\ [\text{“10”}] \end{bmatrix} \ 11\ \ 12\ \ 13\ \ 14\ \ 15\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^T$

$$subs(d, ve) = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 10 \\ \text{“10”} \end{bmatrix} & 11 \end{bmatrix} & 13 & & & 15 & \\ & & & & & \\ & 12 & & 14 & \begin{bmatrix} 0 & \begin{bmatrix} & \begin{bmatrix} 0 & \end{bmatrix} & \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & [0\ \ 0] \end{bmatrix} \\ & & & \begin{bmatrix} [0\ \ 0] & & 0 & \end{bmatrix} & \end{bmatrix} \end{bmatrix}$$

меньше чем листьев в ma, то при исчерпании we замещения прекращаются, и остальные листья ma остаются без изменений. В контрольных примерах рассмотрены как случаи, когда у вектора-донора we все элементы скаляры или строки, так и случаи, когда в we имеются элементы-массивы.

5. Заключение

Из представленного материала вытекает, что для решения задач с гнездовыми массивами во многих случаях полезными могут быть рекурсивные функции. При определенном навыке их написание труда не составляет, а получаемые при этом программы весьма лаконичны и легко сопровождаемы.

СПИСОК ЦИТИРОВАННОЙ ЛИТЕРАТУРЫ

1. Есаян А. Р. Обучение алгоритмизации на основе рекурсии. Тула: Изд. ТГПУ, 2001, с. 215

2. Brent Maxfield, P. E. Essential PTC Mathcad Prime 3.0. A Guide for New and Cur-rent Users, New York, Academic Press is an imprint of Elsevier, Nov. 11, 2013, p. 563

3. Hans Wessenlingh and Hans de Waard. Calculate & Communicate with Mathcad Prime 3.0, Delft Academic Press, The Netherlands, First edition 2014.

REFERENCES

1. Esayan, A. R. 2001, "*Obuchenie algoritmizacii na osnove rekursii.*", [Teaching algorithmization based on recursion], TGPU, Tula, 215 pp. (Russian)

2. Brent Maxfield, P. E. 2013, *Essential PTC Mathcad Prime 3.0. A Guide for New and Cur-rent Users*, Academic Press, New York.

3. Wessenlingh, H. & de Waard, H. 2014, *Calculate & Communicate with Mathcad Prime 3.0*, Delft Academic Press, The Netherlands.

Тульский государственный педагогический университет им. Л. Н. Толстого.
Поступило 26.03.2015