ЧЕБЫШЕВСКИЙ СБОРНИК
Том 16 Выпуск 3 (2015)

ВЕКТОРИЗАЦИЯ И ГНЕЗДОВЫЕ МАССИВЫ

А. Р. Есаян, А. В. Якушин (Тула)

Аннотация

В *PTC Mathcad*, да и в прежних версиях *Mathcad*, для числовых и символьных вычислений предложена специальный оператор векторизации, с помощью которого можно выполнять многие встроенные и некоторые пользовательские функции одной переменной над каждым скалярным или строковым элементом простых или гнездовых (вложенных) массивов. Этот оператор выглядит в виде направленной слева направо стрелки над выражением. Операцию векторизации можно применять и к встроенным функциям нескольких переменных, но только над простыми массивами со скалярными или строковыми элементами. Итак, подчеркнем, что для встроенных функций от одной или нескольких переменных операция векторизации в случае гнездовых массивов может быть реализована далеко не всегда. А для пользовательских функций она, как правило, не реализуется даже для простых массивов.

В статье сняты все упомянутые ограничения, то есть построены аналоги операции векторизации для любых встроенных или пользовательских функций от одной или нескольких переменных при простых или гнездовых массивах. Предложены компактные рекурсивные функции, выполняющие роль оператора векторизации. Рассмотрено два возможных подхода к решению данной задачи. При первом подходе для функций g от n переменных строятся отдельные рекурсивные программы-функции $F1$, $F2$, $F3$, ..., реализующие векторизацию соответственно при $n = 1, 2, 3, \ldots$. При втором подходе для функции g от n переменных создается единая при любых $n = 1, 2, ...$ программа-функция F, выполняющую роль оператора векторизации. В связи с задачей векторизации гнездовых массивов сформулированы некоторые вспомогательные задачи и для них предложены решения в виде рекурсивных функций.

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

Библиография: 3 названия

VECTORIZATION AND NESTED ARRAYS

A. R. Esayan, A. V. Yakushin (Tula)

Abstract

In *PTC Mathcad*, and in previous versions of *Mathcad*, for numeric and symbolic computation proposed special vectorization operation, which can be used to perform many built-in and some custom functions of one variable over each scalar element or string element simple or nested arrays. This operator looks in the form of directed arrows from left to right over expression. The operation of the vectorization can be applied to built-in functions of several variables, but only over a simple array with a scalar or string elements. So, we emphasize that for built-in functions of one or several variables vectorization operation in the case of nested arrays can be implemented not always. And for user-defined functions, it is usually not implemented even for simple arrays.

In the article removed all these constraints and are constructed analogues operation vectorization for any built-in or user-defined functions from one or more variables over simple or nested arrays. There are proposed compact recursive functions that perform the role of the vectorization operator. We considered two possible approaches to solving this problem. When you first approach for functions g of n variables are constructed separate recursive programs-functions $F1$, $F2$, $F3$, ..., implement vectorization respectively for $n = 1, 2, 3, ...$. The second approach for a function g from n variables creates a single for any $n = 1, 2, ...$ program-function F that performs the role of the vectorization operator. In connection with the problem of vectorization for nested arrays formulated some auxiliary problem and were proposed solutions in a form of recursive functions.

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

Bibliography: 3 titles.

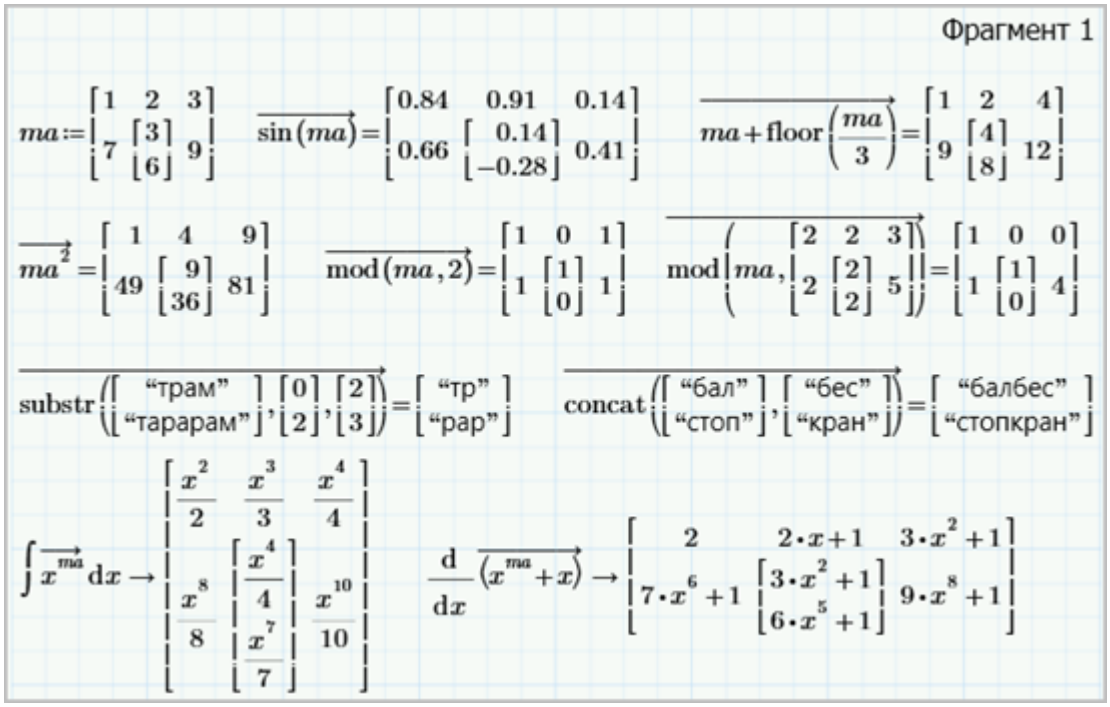
1. Введение

В *PTC Mathcad* и в прежних версиях *Mathcad* имеется необычная операция, называемая операцией векторизации [2, 3]. С ее помощью можно выполнять многие встроенные функции g одной переменной над простым или гнездовым массивом ma, точнее над каждым скалярным или строковым элементом такого массива. Оператор векторизации выглядит в виде направленной слева направо стрелки над выражением. Реализовать операцию векторизации можно так: нажать ключ $Ctrl + Shift + \hat{}$ (или щелкнуть на ленте по кнопке $\vec{V}$), ввести выражение $g(ma)$ и нажать клавишу "=". В документе появиться запись вида $\overrightarrow{g(ma)} = M$, где M — массив такой же структуры, что ma, и для любой пары (a, c) соответствующих друг другу скалярных или строковых элементов $a \in ma$ и $c \in M$ будет выполняться соотношение $g(a) = c$. Вместо функций g в операции векторизации могут участвовать выражения от ma.

Операцию векторизации можно применять и к встроенным функциям g нескольких переменных, но только над простыми массивами со скалярными или строковыми элементами. Например, пусть g — функция двух переменных и

ma, mb — простые массивы одинаковой структуры, то есть количества строк и столбцов в них равны. В данном случае векторизацию g можно реализовать так: нажать ключ $Ctrl + Shift + \hat{}$, ввести выражение $g(ma, mb)$ и нажать клавишу "=". В документе появляется запись вида $\overrightarrow{g(ma,mb)} = M$, где M — массив такой же структуры, что и ma, причем для любой тройки (a, b, m) соответствующих друг другу элементов $a \in ma$, $b \in mb$ и $c \in M$ будет выполняться соотношение $g(a, b) = c$.

Векторизация используется и в символьных вычислениях ($\rightarrow$). Причем для таких вычислений некоторые операторы могут применяться к массивам непосредственно, без специального оператора векторизации (вычисление неопределенных и определенных интегралов, нахождение производных, ...). Фактически, векторизацией во всех ее проявлениях организуются скрытые циклические вычисления. Примеры вычислений с оператором векторизации для гнездовых массивов приведены на фрагменте 1. Там в первых двух строках операция векторизации, применена к встроенной функции синуса $sin(x)$, к выражению $ma + floor(x/3)$, а также к встроенной функции двух переменных $mod(x, y)$. В третьей строке демонстрируется использование векторизации со строковыми функциями. В последней строке по матрице ma с помощью операции векторизации строятся соответственно матрицы $\overrightarrow{x^{ma}}$ и $\overrightarrow{x^{ma+x}}$. При этом последующие интегрирование и дифференцирование автоматически реализуется как векторные операции.



Подчеркнем, что для встроенных функций от одной или нескольких переменных операция векторизации в случае гнездовых массивов может быть реализована далеко не всегда. А для пользовательских функций она, как правило,

не выполняется даже для простых массивов. В статье показывается, как снять все эти ограничения. А именно, с помощью рекурсии создан аналог операции векторизации для любых встроенных или пользовательских функций g от одной или нескольких переменных при простых или гнездовых массивах [1]. При этом рассмотрены два возможных подхода к решению поставленной задачи.

2. Два подхода к векторизации

Подход 1. Для функций g от n переменных для конкретных $n = 1, 2, 3, \ldots$ будем строить отдельные рекурсивные программы-функции $F1$, $F2$, $F3, \ldots$, выполняющие роль операторов векторизации:

$$F1(ma1, g) \text{ играет роль } \overrightarrow{g(ma1)},$$
$$F1(ma1, ma2, g) \text{ играет роль } \overrightarrow{g(ma1, ma2)},$$
$$F1(ma1, ma2, ma3, g) \text{ играет роль } \overrightarrow{g(ma1, ma2, ma3)} \text{ и т. д.}$$

В примерах 1 фрагмента 2 показаны возможные варианты функций $F1$, $F2$, $F3$ и обращения к ним. Ни в одном из них использование встроенной векторизации невозможно. Кратко остановимся на описании рекурсивных функций $F1$, $F2$, $F3$:

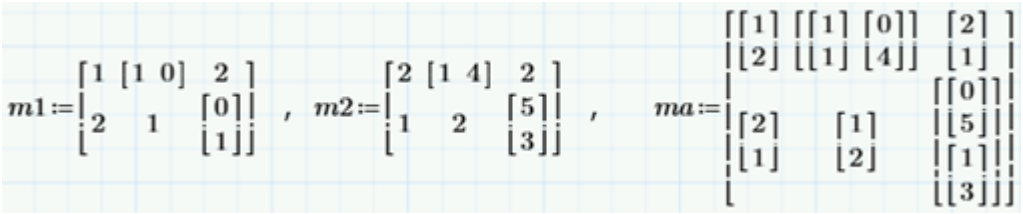
- $F1$ устроена так. При вычислении $F1(ma, g)$ создается матрица B такой же структуры, что и ma, и в двойном цикле for для каждого элемента верхнего уровня $a = ma_{i,j}$ из ma реализуются следующие действия. Если a не является массивом, то элемент $B_{i,j}$ полагается равным $g(a)$, иначе — рекурсивному обращению $F1(a, g)$. Во втором случае $F1$ таким же способом начинает выполняться для найденного подмассива a и т. д. Все это и обеспечивает формирование матрицы B, скалярные или строковые элементы которой (листья), являются результатом применения g к соответствующим листьям исходной матрицы ma;
- $F2$ устроена так. При вычислении $F2(ma, mb, g)$ создается матрица B такой же структуры, что и ma, и в двойном цикле for для каждой пары соответствующих элементов верхнего уровня $a = ma_{i,j} \in ma$ и $b = mb_{i,j} \in mb$ реализуются следующие действия. Если a не массив, то элемент $B_{i,j}$ полагается равным $g(a, b)$, иначе — рекурсивному обращению $F2(a, b, g)$. Во втором случае $F2$ таким же способом начинает выполняться для найденной пары подмассивов a и b и т. д. Все это и обеспечивает формирование матрицы B, листья которой являются результатом применения g к паре соответствующих листьев исходных матриц ma и mb;
- $F3$ устроена аналогично $F2$, только здесь вычисляется $F3(ma, mb, mc, g)$ и $B_{i,j}$ полагается равным $g(a, b, c)$ или рекурсивному обращению $F3(a,b, c, g)$, где a, b и c — соответствующие друг другу элементы матриц ma, mb и mc.

Подход 2. Для функции g n переменных создадим единую при любых $n = 1, 2, ...$ программу-функцию F, выполняющую роль оператора векторизации. При этом:

a) если g — пользовательская функция n переменных, то определять ее будем как функцию компонентов вектора v: $g(v) := g(v_0, v_1, ..., v_{n-1})$;

b) если $g(x, y, ...)$ — встроенная функция n переменных, то ее определение будем подправлять, переписывая его в виде: $func(v) := g(v_0, v_1, ..., v_{n-1})$, где $func$ — новое имя функции. Например, встроенная функция трех переменных $substr(st, n, m)$ должна быть записана, например, так $qwe(v) := substr(v_0, v_1, v_2)$;

c) если, при первом подходе, g и F должны применяться к n матрицам $m1$, $m2$, ... одинаковой структуры, то в данном случае вместо $m1$, $m2$, ... должна использоваться одна матрица ma, получаемая заменой в копии $m1$ каждого листа вектором из соответствующих друг другу последовательных листьев матриц $m1$, $m2$, ... Например, вместо матриц $m1$ и $m2$ следует использовать приведенную справа от них матрицу ma:



Второй подход к решению поставленной задачи, при соблюдении пунктов a, b и c, реализован в примере 2 фрагмента 2 парой рекурсивных функций F и hei, где F — головная функция, hei — вспомогательная функция для вычисления высоты массива. Опишем действие функции F. При вычислении $F(ma, g)$ создается матрица B такой же структуры, что и ma, и в двойном цикле for для каждого элемента верхнего уровня $b = ma_{i,j}$ из ma реализуются следующие действия. Если высота b равна 1, то есть b состоит только из листьев, то элемент $B_{i,j}$ полагается равным $g(b)$, иначе — рекурсивному обращению $F(b, g)$. Во втором случае F таким же способом начинает выполняться для найденного подмассива b и т. д. Все это и обеспечивает правильное формирование выводимой матрицы B.

3. Дополнительные средства для векторизпции

В связи с задачей векторизации гнездовых массивов полезными могут быть вспомогательные рекурсивные функции $assem1$, $assem2$, $assem$, $reas$ и $tree$ решения сформулированных ниже задач A, B и C.

Задача A. По n простым или гнездовым массивам одинаковой структуры $m1$, $m2$, ... со скалярными или строковыми листьями создать массив ma следующим образом: в копии $m1$ на месте каждого листа поместить вектор, последовательные компоненты которого есть соответствующие друг другу листья массивов $m1$, $m2$, ...

Решение. На фрагменте 2 решение поставленной задачи для частных случаев предлагается в виде рекурсивных функций $assem1$ ($n = 1$) и $assem2$ ($n = 2$). По таким же схемам можно было бы построить функцию для решения задачи A при любом фиксированном n. Желательно, конечно, иметь одну функцию для решения A при любом натуральном n. Представлена и такая рекурсивная функция — $assem$. Правда в ней аргументы, в отличие от $assem1$ и $assem2$, задаются не отдельными массивами, а вектором массивов. Кратко остановимся на описании функций $assem1$, $assem2$ и $assem$.

Функция $assem1(m1, g)$ устроена так. Создается массив B, являющийся копией $m1$. Далее, в двойном цикле for организуется пробежка по элементам $m1$. При этом из очередного элемента $m1_{i,j}$ создается вектор ve : $ve \leftarrow [m1_{i,j}]$. Далее, если ve_0 не является массивом, то $B_{i,j}$ полагается равным ve, иначе — рекурсивному обращению $assem1(ve_0)$. Все это и обеспечивает формирование матрицы B, полученной заменой в $m1$ листьев одноэлементными векторами из этих листьев.

Функция $assem2(m1, m2, g)$ устроена так. Создается массив B, являющийся копией $m1$. Далее, в двойном цикле for организуется пробежка по элементам $m1$. При этом из очередной пары соответствующих друг другу элементов $m1_{i,j}$ и $m2_{i,j}$ массивов $m1$ и $m2$ создается вектор ve: $ve \leftarrow [m1_{i,j}\ m2_{i,j}]^T$. Если ve_0 не является массивом, то $B_{i,j}$ полагается равным ve, в противном случае — рекурсивному обращению $assem1(ve_0, ve_1)$. Все это и обеспечивает формирование матрицы B, получаемой заменой в $m1$ листьев двухэлементными векторами с компонентами из соответствующих листьев $m1$ и $m2$.

Функция $assem(ve, g)$ устроена так. Создается массив B, являющийся копией ve_0 и в двойном цикле for организуется пробежка по элементам ve_0. При этом, из последовательных элементов $(ve_k)_{i,j}\,(k = 0, 1, ..., last(ve))$ создается вектор w: $w \leftarrow \left[(ve_0)_{i,j}\ (ve_1)_{i,j}\ ...\ (ve_{last(ve)})_{i,j}\right]^T$. Далее, если w_0 не является массивом, то $B_{i,j}$ полагается равным w, в противном случае — рекурсивному обращению $assem(w)$. Все это и обеспечивает формирование матрицы B, получаемой заменой в $m1$ листьев вектором с компонентами из соответствующих друг другу листьев массивов $m1$, $m2$, ...

Задача B. Написать функцию $reas$, обратную к функции $assem$ (см. решение задачи A). Иными словами, результат выполнения функции $assem(ve)$ по $reas$ должен преобразовываться в ve: $reas(assem(ve)) = ve$, где ve — вектор, компоненты которого есть массивы одной и той же структуры. И обратно, результат выполнения функции $reas(ma)$ по $assem$ должен преобразовываться в

ma: $assem(reas(ma)) = ma$, где ma — массив, у которого все подмассивы, расположенные на 1 уровень выше листьев, являются векторами одной и той же длины.

Решение. На фрагменте 2 представлено одно из возможных рекурсивных решений поставленной задачи. Там же приведены и вычисления по этой функции. Ее описание достаточно громоздкое, и поэтому приводить его не будем.

Задача C. Сгенерировать случайный простой или гнездовой массив высоты не более n (при равномерном распределении). Если высота полученного массива оказывается больше n, то должно выводиться сообщение "Высота $> n$". Аварийный выход при переполнении стека рекурсивных вызовов должен быть обработан. При решении задачи можно зафиксировать $n = 5$, потому, что при $n > 5$ полученные массивы обычно очень громоздкие и в развернутом виде для вывода в документе не приспособлены.

Решение. Задача решается функциями $tree(p)$ и $rtree(p)$, представленными на фрагменте 2, где p — вероятность того, что по $rnd(1)$ будет сгенерировано случайное число x ($0 \le x < p < 1$). Функция $tree(p)$ является головной. При вычислениях она обращается к рекурсивным функциям hei и $rtree$.

В $tree$ с помощью оператора $try\ \ldots\ on\ error$ реализуется обработка прерывания. Если при обращении к $rtree(p)$ происходит прерывание по переполнению стека рекурсивных вызовов, то в $tree$ выводится соответствующее сообщение и вычисления прекращаются. В противном случае, если в сгенерированном массиве B высота оказывается больше 5, то также происходит вывод соответствующего сообщения и вычисления прекращаются, иначе выводится сгенерированный гнездовой массив B.

Функция $rtree(p)$ выполняется так. Случайным образом для генерации очередного подмассива B выбирается количество строк (от 1 до 3) и количество столбцов (от 1 до 3). В двойном цикле for следующим образом формируются элементы этого подмассива. Генерируется случайное целое число $x : 0 \le x = floor(rnd(9)) < 9$. При $rnd(1) \ge p$ это число становится значением очередного элемента $B_{i,j}$ подмассива B. При $rnd(1) < p$ происходит рекурсивный вызов функции $rtree(p)$. Все это и обеспечивает генерацию гнездового массива высоты не более 5.

Замечания. 1. В двойном цикле for функции $rtree$ верхние границы индексов i и j равны $floor(rnd(3))$. Это обеспечивает генерирование гнездовых массивов, у которых подмассивы на любом уровне вложенности могут иметь случайный размер $n \times m$, где $1 \le n, m \le 3$. При необходимости, эти границы можно изменить на $floor(rnd(n))$ и $floor(rnd(m))$, причем величины n и m сделать дополнительными аргументами $rtree$ и $tree$, а при обращении к $tree$ инициализировать их любыми целыми неотрицательными значениями.

2. В теле двойного цикла for функции $rtree$ выражение $floor(rnd(9))$ обеспечивает листья генерируемого массива целочисленными значениями, располо-

женными между 0 и 8. Эти значения можно сделать действительными, убрав $floor$, и привести к любому другому диапазону $[a, b)$ $(a < b)$, используя выражение $a + rnd(b - a)$. Если необходимо генерировать целочисленные значения диапазона $[a, b]$ $(a \le b, a, b$ — целые), то выражение можно задать в виде $a + floor(rnd(b - a + 1))$.

3. Встроенная функции *Seed*, используемая при вычислениях с *tree* и *rtree*, инициализирует генератор случайных чисел начальным значением. Если это значение зафиксировать, то генерируемая последовательность при любом новом запуске вычислений будет одна и та же. Этот факт используется при отладке программ. Напомним, что вычисления (перевычисления) в документе запускаются или ключом $Ctrl + F5$, или командой ленты "Расчет/Элементы управления/Рассчитать" .

4. Необычность рекурсии в $rtree(p)$ состоит в том, что функция обращается сама к себе с тем же самым значением аргумента p, а проблема перемещения по рекурсивным уровням решается не за счет изменения её аргумента, а за счет анализа полученного значения от датчика случайных чисел. Это достаточно типичный случай для "вероятностной" рекурсии.

5. По головной программе-функции *tree* могут выводиться сообщения "*Высота* > 5" и "*Переполнение стека рекурсивных вызовов*". Но нас, по-видимому, интересуют лишь сгенерированные массивы, а не сообщения. Поэтому, прекращение вычислений явление нежелательное и, следовало бы заменить вывод любого сообщения попыткой повторного генерирования массива. Это и сделано во втором варианте головной программы-функции *setree*, которая, в отличие от *tree*, уже рекурсивная. В ней в блоке *on error* вместо сообщения о переполнении стека реализуется обращение к себе, то есть рекурсивные вычисления. Далее, в блоке *try* вместо обращения к *rtree* с возможным сообщением о неподходящей высоте массива реализуются циклические обращения к *rtree*.

4. Заключение

Для пакета инженерных и научных вычислений *PTC Mathcad* [2,3] предложены компактные рекурсивные программы-функции, реализующие операцию векторизации и применимые для любых встроенных и пользовательских функций от одной или нескольких переменных при простых или гнездовых массивах. Эти функции снимают все ограничения, присущие встроенному оператору векторизации.

Фрагмент 2 (1,2,3,4,5)

Примеры 1. Векторизация функции g n-переменных над гнедовыми массивами $ma1$, $ma2$, ... одинаковой структуры (отдельно для n=1,2 и 3).

n=1. $F1(ma,g):=$
```
B ← ma
for i ∈ 0..rows(ma)−1
    for j ∈ 0..cols(ma)−1
        a ← ma_{i,j}
        B_{i,j} ← if(IsArray(a), F1(a,g), g(a))
B
```

$b:=\begin{bmatrix}1 & -3 & 7\\ 7 & 6 & [7\ 5]\end{bmatrix}$ $we:=\begin{bmatrix}\text{“один”}\\ [\text{“два”}\ \ 3333]\end{bmatrix}$ $F1(b,\text{num2str})=\begin{bmatrix}\text{“1”} & \text{“−3”} & \text{“7”}\\ \text{“7”} & \text{“6”} & [\text{“7”}\ \text{“5”}]\end{bmatrix}$

$F1(b,\text{sign})=\begin{bmatrix}1 & -1 & 1\\ 1 & 1 & [1\ 1]\end{bmatrix}$ $F1(we,\text{IsString})=\begin{bmatrix}1\\ [1\ 0]\end{bmatrix}$ $F1(we,\text{IsScalar})=\begin{bmatrix}0\\ [0\ 1]\end{bmatrix}$

n=2. $F2(ma,mb,g):=$
```
B ← ma
for i ∈ 0..rows(ma)−1
    for j ∈ 0..cols(ma)−1
        [a ← ma_{i,j}  b ← mb_{i,j}]
        B_{i,j} ← if(IsArray(a), F2(a,b,g), g(a,b))
B
```

$c1:=\begin{bmatrix}[7\ 9] & 3 & & 7\\ & & \begin{bmatrix}\ & \begin{bmatrix}2 & \begin{bmatrix}7 & 2 & 3\\ 4 & 5 & [77\ \ 99]\end{bmatrix}\end{bmatrix}\end{bmatrix} & \\ 4 & 7 & [4 & 7\end{bmatrix}\end{bmatrix}$ $c2:=\begin{bmatrix}[7\ 1] & 1 & & 7\\ & & \begin{bmatrix}\ & \begin{bmatrix}2 & \begin{bmatrix}7 & 2 & 3\\ 1 & 5 & [7\ 9]\end{bmatrix}\end{bmatrix}\end{bmatrix} & \\ 0 & 7 & [4 & 7\end{bmatrix}\end{bmatrix}$

$F2(c1,c2,\text{gcd})=\begin{bmatrix}[7\ 1] & 1 & & 7\\ & & \begin{bmatrix}\ & \begin{bmatrix}2 & \begin{bmatrix}7 & 2 & 3\\ 1 & 5 & [7\ 9]\end{bmatrix}\end{bmatrix}\end{bmatrix} & \\ 4 & 7 & [4 & 7\end{bmatrix}\end{bmatrix}$

$x:=\begin{bmatrix}\text{“бал”}\\ [\text{“123”}\ \text{“век”}]\end{bmatrix}$ $y:=\begin{bmatrix}\text{“бес”}\\ [\text{“45”}\ \text{“тор”}]\end{bmatrix}$ $F2(x,y,\text{concat})=\begin{bmatrix}\text{“балбес”}\\ [\text{“12345”}\ \text{“вектор”}]\end{bmatrix}$

n=3. $F3(ma,mb,mc,g):=$
```
B ← ma
for i ∈ 0..rows(ma)−1
    for j ∈ 0..cols(ma)−1
        [a ← ma_{i,j}  b ← mb_{i,j}  c ← mc_{i,j}]
        B_{i,j} ← if(IsArray(a), F3(a,b,c,g), g(a,b,c))
B
```

Фрагмент 2 (1,**2**,3,4,5)

$$func(x,y,z) := \text{floor}\left(\frac{x}{2}\right) + \text{trunc}\left(\frac{y}{3}\right) - \text{ceil}\left(\frac{y}{x}\right)$$

$$F3(c1, c2, c2+1, func) = \begin{bmatrix} [[4\ \ 3]\ \ 0 & & & 4 & \\ \begin{bmatrix} & & \begin{bmatrix} & \begin{bmatrix} 4 & 0 & & 1 \end{bmatrix} \end{bmatrix} \\ 2 & 4 & \begin{bmatrix} 0 & \begin{bmatrix} 0 & \begin{bmatrix} 1 & 2 & [39 & 51] \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ & & \begin{bmatrix} & \begin{bmatrix} 2 & & 4 \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Пример 2. Векторизация функции *g* *n*-переменных над гнездовыми массивами *ma*1, *ma*2, ... (создается единая программа-функция для *n*=1, 2, ...)

```
hei(ma) := h ← −1
           if IsArray(ma)
             for b ∈ ma
               h ← max(h, hei(b))
           h + 1
```

подсчет максимальной высоты (глубины вложенности) гнездоваго массива *ma*

```
F(ma, g) := B ← ma
            for i ∈ 0..rows(ma) − 1
              for j ∈ 0..cols(ma) − 1
                b ← ma_{i,j}
                B_{i,j} ← if(hei(b) = 1, g(b), F(b, g))
            B
```

1)

$$a := \begin{bmatrix} [[0.3] & [-.7] \\ [6] & \begin{bmatrix} \begin{bmatrix} 6 \\ 7 \end{bmatrix} & [5.5] \end{bmatrix} \end{bmatrix}$$

$$myround(v) := \text{round}\left(v_0\right)$$

$$F(a, myround) = \begin{bmatrix} 0 & -1 \\ 6 & [1\ \ 6] \end{bmatrix}$$

2)

$$ma := \begin{bmatrix} [[45] & [81] & [18] \\ [90] & [9] & [4] \\ [45] & [45] & [[45] & [20]] \\ [90] & [90] & [[90] & [90]] \end{bmatrix}$$

$$mgcd(v) := \gcd\left(v_0, v_1\right)$$

$$F(ma, mgcd) = \begin{bmatrix} 45 & 9 & 2 \\ 45 & 45 & [45\ \ 10] \end{bmatrix}$$

3)

$$w := \begin{bmatrix} [[\text{“вишня”} & [\text{“яблоко”} & & [\text{“груша”} & \\ [0] & [1] & & [2] & \\ [2] & [3] & & [6] & \\ [\text{“слива”} & [\text{“персик”} & [[\text{“апельсин”} & [\text{“мандарин”}]] & \\ [0] & [0] & [3] & [4] & \\ [3] & [3] & [2] & [2] & \end{bmatrix}$$

$$mysubstr(v) := \text{substr}\left(v_0, v_1, v_2\right)$$

$$F(w, mysubstr) = \begin{bmatrix} \text{“ви”} & \text{“бло”} & \text{“уша”} \\ \text{“сли”} & \text{“пер”} & [\text{“ль”}\ \ \text{“ар”}] \end{bmatrix}$$

Задача А. По *n* простым или гнездовым массивам одинаковой структуры *m*1, *m*2, ... со скалярными или строковыми листьями создать массив *ma* следующим образом: в копии *m*1 на месте каждого листа поместить вектор, последовательные компоненты которого есть соответствующие друг другу листья массивов *m*1, *m*2, ...

Фрагмент 2 (1,2,3,4,5)

n=1.
$$assem1(m1) := \left\| \begin{array}{l} B \leftarrow m1 \\ \text{for } i \in 0..\text{rows}(m1)-1 \\ \quad \text{for } j \in 0..\text{cols}(m1)-1 \\ \quad\quad ve \leftarrow \left[m1_{i,j} \right] \\ \quad\quad B_{i,j} \leftarrow \text{if}\left(\text{IsArray}\left(ve_0\right), assem1\left(ve_0\right), ve\right) \\ B \end{array} \right.$$

n=2.
$$assem2(m1,m2) := \left\| \begin{array}{l} B \leftarrow m1 \\ \text{for } i \in 0..\text{rows}(m1)-1 \\ \quad \text{for } j \in 0..\text{cols}(m1)-1 \\ \quad\quad ve \leftarrow \begin{bmatrix} m1_{i,j} \\ m2_{i,j} \end{bmatrix} \\ \quad\quad B_{i,j} \leftarrow \text{if}\left(\text{IsArray}\left(ve_0\right), assem2\left(ve_0, ve_1\right), ve\right) \\ B \end{array} \right.$$

Общий случай. Здесь *ve* - вектор с *n* аргументами

$$assem(ve) := \left\| \begin{array}{l} B \leftarrow ve_0 \\ \text{for } i \in 0..\text{rows}\left(ve_0\right)-1 \\ \quad \text{for } j \in 0..\text{cols}\left(ve_0\right)-1 \\ \quad\quad \text{for } k \in 0..\text{last}(ve) \\ \quad\quad\quad w_k \leftarrow \left(ve_k\right)_{i,j} \\ \quad\quad B_{i,j} \leftarrow \text{if}\left(\text{IsArray}\left(w_0\right), assem(w), w\right) \\ B \end{array} \right.$$

$$b1 := \begin{bmatrix} 3 & 7 \\ 6 & [7\ \ 5] \end{bmatrix} \qquad b2 := \begin{bmatrix} \text{“2”} & \text{“3”} \\ \text{“1”} & [\text{“5”}\ \ \text{“6”}] \end{bmatrix} \qquad b3 := \begin{bmatrix} 0 & 2 \\ 1 & [3\ \ 54] \end{bmatrix}$$

1) $$assem1(b1) = \begin{bmatrix} [3] & [7] \\ [6] & [[7]\ \ [5]] \end{bmatrix} \qquad assem1(b2) = \begin{bmatrix} [\text{“2”}] & [\text{“3”}] \\ [\text{“1”}] & [[\text{“5”}]\ \ [\text{“6”}]] \end{bmatrix}$$

$$assem([b1]) = \begin{bmatrix} [3] & [7] \\ [6] & [[7]\ \ [5]] \end{bmatrix} \qquad assem1(b1) = assem([b1]) = 1$$

2) $$assem\begin{pmatrix} [b1] \\ [b2] \end{pmatrix} = \begin{bmatrix} [3] & [7] \\ [\text{“2”}] & [\text{“3”}] \\ [6] & [[7]\ \ [5]] \\ [\text{“1”}] & [[\text{“5”}]\ \ [\text{“6”}]] \end{bmatrix} \qquad assem\begin{pmatrix} [b1] \\ [b2] \\ [b3] \end{pmatrix} = \begin{bmatrix} [3] & [7] \\ [\text{“2”}] & [\text{“3”}] \\ [0] & [2] \\ [6] & [[7]\ \ [5]] \\ [\text{“1”}] & [[\text{“5”}]\ \ [\text{“6”}]] \\ [1] & [[3]\ \ [54]] \end{bmatrix}$$

$$assem2(b1,b2) = \begin{bmatrix} [3] & [7] \\ [\text{“2”}] & [\text{“3”}] \\ [6] & [[7]\ \ [5]] \\ [\text{“1”}] & [[\text{“5”}]\ \ [\text{“6”}]] \end{bmatrix} \qquad assem2(c1,c2) = assem\begin{pmatrix} [c1] \\ [c2] \end{pmatrix} = 1$$

Фрагмент 2 (1,2,3,**4**,5)

Задача B. Написать рекурсивную функцию *reas*, обратную к функции *assem* (см. решение задачи *A*). Иными словами, результат выполнения функции *assem*(*ve*) по *reas* должен преобразовываться в *ve*: *reas*(*assem*(*ve*))=*ve*, где *ve* – вектор, компоненты которого есть массивы одной и той же структуры. И обратно, результат выполнения функции *reas*(*ma*) по *assem* должен преобразовываться в *ma*: *assem*(*reas*(*ma*))=*ma*, где *ma* – массив, у которого все подмассивы, расположенные на 1 уровень выше листьев, являются векторами одной и той же длины.

```
reas(ma) := tt ← ma
            while hei(tt) ≠ 1
              tt ← tt_{0,0}
            for k ∈ 0..last(tt)
              ve_k ← ma
            for i ∈ 0..rows(ma)−1
              for j ∈ 0..cols(ma)−1
                b ← ma_{i,j}
                r ← if(hei(b) = 1, b, reas(b))
                for k ∈ 0..last(ve)
                  (ve_k)_{i,j} ← r_k
            ve
```

$$a := \begin{bmatrix} [3] & [7] \\ [6] & [[11]\ [6]] \end{bmatrix}$$

$$reas(a) = \begin{bmatrix} [3 & 7] \\ [6 & [11\ 6]] \end{bmatrix}$$

$$b := \begin{bmatrix} [3] & [7] \\ [\text{“2”}] & [\text{“3”}] \\ [6] & [[7] & [5]] \\ [\text{“1”}] & [\text{“5”}] & [\text{“6”}] \end{bmatrix}$$

$$reas(b) = \begin{bmatrix} [3 & 7] \\ [6 & [7\ 5]] \\ [\text{“2”} & \text{“3”}] \\ [\text{“1”} & [\text{“5”}\ \text{“6”}] \end{bmatrix}$$

$$d := \begin{bmatrix} [2] \\ [3] \\ [4] \end{bmatrix}$$

$$reas(d) = \begin{bmatrix} [[2] & [2 & [3 & [[7\ 2\ 3]] \\ [3] & [4\ 5\ [7\ 0]]] \\ [4] & [4 & 7]] \\ [[3 & [3 & [8\ 3\ \text{“4”}]] \\ [5\ \text{“6”}\ [8\ 1]]] \\ [5 & 8]] \\ [[4 & [4 & [9\ 4\ 5]] \\ [6\ 7\ [9\ 2]]] \\ [6 & 9]] \end{bmatrix}$$

$$assem(reas(d)) = d = 1 \qquad ve := reas(d) \qquad reas(assem(ve)) = ve = 1$$

Задача C. Сгенерировать случайный гнездовой массив высоты не более 5

```
rtree(p) := [n  m] ← [floor(rnd(3))  floor(rnd(3))]
            for i ∈ 0..n
              for j ∈ 0..m
                B_{i,j} ← if(rnd(1) < p, rtree(p), floor(rnd(9)))
            B
```

Фрагмент 2 (1,2,3,4,5)

```
tree(p) := try
             B ← rtree(p)
             if(hei(B) > 5, "Высота>5", B)
           on error
             "Переполнение стека рекурсивных вызовов"
```

Вариант головной программы с двумя сообщениями:
1) "Высота>5";
2) "Переполнение стека рекурсивных вызовов"

```
setree(p) := try
               B ← rtree(p)
               while hei(B) > 5
                 B ← rtree(p)
             on error
               setree(p)
```

Вариант головной программы без вывода сообщений

1) $y := \mathrm{Seed}(73190595)$ $p := 0.2$ 2 последовательных вызова tree

$$tree(p) = \begin{bmatrix} & & [3\ 0\ [1\ 8]] \\ 7 & 8 & [6\ 5\ \ 6\ \ ] \\ & & [1\ 6\ \ 1\ \ ] \\ & [4\ 3\ 8] & \\ 3 & [5\ 6\ 6] & 6 \\ & [8\ 3\ 1] & \\ 4 & 7 & 2 \end{bmatrix} \qquad tree(p) = \begin{bmatrix} [[\ \ [\ \ 6\ \ ]\ ] & \\ [5\ \ [[7\ 1]\ 1]] & \\ [\ \ \ [[\ 4\ \ 8]]] & 2 \\ [8\ [[5]\ [3\ 6]\ 8]] & \\ [\ [[4]\ [1\ 7]\ ]] & \\ [\ \ [[5]] & 0 \\ [\ \ [[2]] & \\ [\ \ \ 8 & [1\ 2] \\ [& [6\ 6]] \end{bmatrix}$$

2) $y := \mathrm{Seed}(177185)$ $p := 0.3$ 9 последовательных вызов tree

$$tree(p) = \begin{bmatrix} 0 & 3 \\ 3 & [5] \\ 7 & 6 \end{bmatrix} \qquad tree(p) = \begin{bmatrix} [[\ \ \ [1] & & \\ [\ \ \ [4] & 2 & \\ [\ \ \ [3] & & \\ [[[[0\ 1] & & \\ [[0\ 2] & 2\ 4 & 4\ 3 \\ [\ \ \ 8 & 1\ 6 & 6 \\ [[\ \ \ 0 & 3\ 6] & \\ 5 & & 7 \\ 7 & & 3 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ [2] \\ \\ \\ \\ \\ \end{matrix}$$

$tree(p) = [6\ \ 8]$

$tree(p) =$ "Переполнение стека рекурсивных вызовов"

$tree(p) =$ "Высота>5" $\qquad tree(p) = \begin{bmatrix} & [2] & \\ [0 & 5 & 3] \\ [6 & [6\ 5\ 8] & 2] \end{bmatrix} \qquad tree(p) = [[3\ 5\ 7]\ 4\ 5]$

$tree(p) = [3\ 7\ 7]$ $\qquad tree(p) =$ "Переполнение стека рекурсивных вызовов"

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 21.04.2015