Златопольский Дмитрий Михайлович
Кандидат технических наук, доцент.
Доцент кафедры информатики и прикладной математики Московского городского педагогического университета; организатор и директор Музея истории вычислительной техники.

Вспоминая барона Мюнхгаузена…

«Схватив себя за эту косичку (своего парика – Д. З.), я изо всех сил дёрнул вверх и без большого труда вытащил из болота и себя, и своего коня… Если вы думаете, что это легко, попробуйте проделать это сами».

Барон Мюнхгаузен

Автор привел цитату известного «правдеца» (☺) потому, что в ней – пример использования такого понятия программирования, как «рекурсия». В статье описываются некоторые особенности последней.

Напомним, что «рекурсией» (от латинского *recursio* – возвращение) называют ситуацию в программе, при которой процедура или функция использует (вызывает) как вспомогательную саму себя, а сами такие процедуры (функции) называют «рекурсивными».

В качестве примера приведём функцию для расчёта факториала натурального числа n (факториал числа n, обозначаемый $n!$, равен $1 \times 2 \times 3 \times \ldots \times n$). Такую функцию можно создать, имея в виду, что $n! = (n - 1)! \times n$. На школьном алгоритмическом языке соответствующая функция имеет вид:

```
алг цел Факториал(арг цел n)
нач
  знач := Факториал(n - 1) * n |Рекурсивный вызов функции Факториал
  |Служебное слово знач означает "значение функции"
кон
```

На самом деле эта функция оформлена не по правилам и при выполнении программы появится сообщение об ошибке. Дело в том, что при каждом вызове вспомогательной функции (или процедуры) для неё отводится место в оперативной памяти, которое «освобождает-

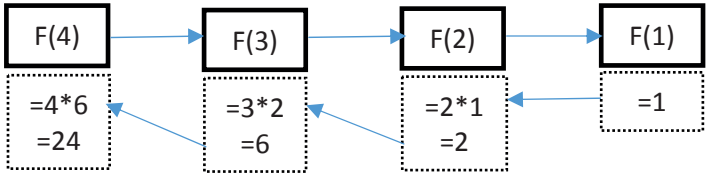
ся» после завершения её (функции/процедуры) работы. Но если во вспомогательной процедуре имеется рекурсия, то вызовы вспомогательных подпрограмм будут продолжаться до тех пор, когда место в памяти будет исчерпано. Чтобы устранить этот недостаток, необходимо так оформлять функции (процедуры), чтобы рекурсивные вызовы осуществлялись по условию, которое когда-то станет ложным. Например, для приведённой функции это условие записывает так:

```
алг цел Факториал(арг цел n)
нач
  если n > 1
    то
      |Рекурсивный вызов функции Факториал
       знач := Факториал(n - 1) * n
    иначе |Известное значение
      знач := 1
  все
кон
```

или

```
алгцел Факториал(арг цел n)
нач
  если n = 1
   то
     знач := 1
   иначе |Рекурсивный вызов функции Факториал
     знач := Факториал(n - 1) * n
  все
кон
```

Рекурсивные вызовы



Обратная передача результатов

Рис. 1

Пример работы программы при вычислении значения факториала числа 4 показан на рис. 1.

Прежде чем приводить ещё один пример использования рекурсии, рассмотрим такую задачу: «Разработать функцию для расчёта k-го члена последовательности Фибоначчи (последовательность Фибоначчи образуют числа 1, 1, 2, 3, 5, 8, 13, …)».

Если вы хотите попробовать создать такую функцию самостоятельно (не используя рекурсию), то не смотрите на приведённый далее вариант:

```
алг цел Фиб(арг цел k)
нач цел очер, пред, предпред, i
пред := 1
предпред := 1
нц для i от 2 до k - 2
очер := пред + предпред
предпред := пред
пред := очер
кц
знач := очер
кон
```

Быстро ли вы получили требуемую функцию? А теперь посмотрите, как просто и логично выглядит рекурсивный вариант функции, определяющей *k*-й член последовательности Фибоначчи:

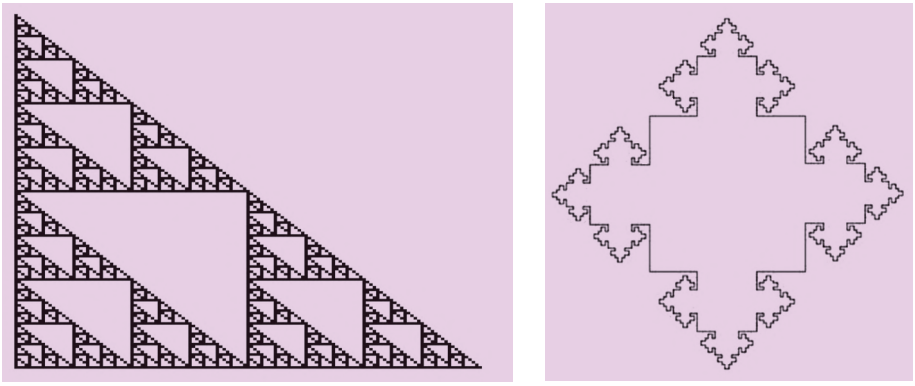
```
алгцел Фиб(арг цел k)
нач
  если k = 1 или k = 2
    то
      знач := 1
    иначе |Рекурсивный вызов функции
      знач := Фиб(k - 2) + Фиб(k - 1)
кон
```

Такое оформление полностью соответствует закону построения последовательности Фибоначчи – очередной элемент последовательности равен сумме двух предыдущих. При нём не требуется применять оператор цикла и думать над последовательностью расчёта значений предшествующего и предшествующего предшествующему членов последовательности.

Справедливости ради следует сказать, что рекурсия – это эффектно, но не всегда эффективно. Например, последняя функция при расчётах вызывается многократно. В процессе вычислений, например, Фиб(17), функция для расчёта Фиб(15) будет вызываться 2 раза, для расчёта Фиб(14) – 3 раза, Фиб(13) – 5 раз, Фиб(12) – 8 раз и так далее. Всего при вычислении Фиб(17) компьютер выполнить более 1000 операций сложения, Фиб(31) – более 1 миллиона, Фиб(45) – более 1 миллиарда операций сложения! [2]. В то же время в нерекурсивном варианте функции в последнем случае происходят всего 43 операции сложения!

Одним из самым ярких примером использования рекурсии является метод сортировки массивов, разработанный профессором Оксфордского университета Чарльзом Хоаром. Этот метод, считающийся самым быстрым из всех известных, основан на рекурсии.

Рекурсия используется в программировании не только при обработке чисел. На рис. 2 показаны два изображения, полученные с помощью рекурсии.



а) б)

Рис. 2

Картинка на рис. 2а получена так. В треугольнике проводятся все средние линии. Тем самым он разбивается на 4 треугольника. К трём из них, примыкающим к вершинам первоначального треугольника, применяются те же действия. Рекурсивная процедура Треугольник, в которой это реализуется, имеет вид:

```
алг Треугольник(арг цел xa, ya, xb, yb, xc, yc, n)
нач цел xp, yp, xq, yq, xr, yr
  если n > 0
    то
    |Координаты середин сторон треугольника
    xp := (xb + xc)/2; yp := (yb + yc)/2
    xq := (xa + xc)/2; yq := (ya + yc)/2
    xp := (xa + xb)/2; yr := (ya + yb)/2
    поз(xp, yp)
    |Рисуем средние линии
    линия(xq, yq)|Стандартная процедура
    линия(xr, yr)
    линия(xp, yp)
    |Рекурсивно вызываем процедуру Треугольник
    Треугольник(xa, ya, xr, yr, xq, yq, n - 1) |для каждой
    Треугольник(xb, yb, xp, yp, xr, yr, n - 1) |из трех
    Треугольник(xc, yc, xq, yq, xp, yp, n - 1) |сторон
  все
кон
```

Основная часть программы:

```
алг Множество_треугольников
нач цел xa, ya, xb, yb, xc, yc
    |Включение графического режима работы
    …
    |Координаты вершин самого большого треугольника
    xc := 0; yc := 0
    xb := максX; yb := максY  (см. рис. 2а)
    xa := 0; ya := максY
    поз(xa, ya)
    |Рисуем самый большой треугольник
    линия(xb, yb)
    линия(xc, yc)
    линия(xa, ya)
    |Начинаем рисовать внутренние треугольники
    Треугольник(xa, ya, xb, yb, xc, yc, 6)
кон
```

Изображение на рис. 2б получено следующим образом. На каждой из сторон внутреннего (самого большого) квадрата нарисованы 3 стороны малого квадрата, на каждой из сторон которого также изображены 3 стороны еще меньшего квадрата и так далее. Процедура, которая выполняет соответствующие действия на некотором отрезке с координатами концов xa, ya, xb, yb, может быть оформлена следующим образом:

```
алг Сторона(арг цел xa, ya, xb, yb, n, вещ k)
нач цел xp, yp, xq, yq, xr, yr, xs, ys, dx, dy
   |xp, yp, xq, yq, xr, yr, xs, ys – координаты вершин
   |малого квадрата
   если n = 0
      то
        линия(xb, yb)
      иначе
        dx := 0.5 * (1 – k) * (xb – xa) |k – коэф. уменьшения
        dy := 0.5 * (1 – k) * (yb - ya) |размера квадратов
        |Координаты фигуры, изображаемой на отрезке ab
        xp := xa + dx; yp := ya + dy
        xs := xb – dx; ys := yb – dy
        xq := xp + (ys – yp); yq := yp – (xs – xp)
        xr := xq + (xs – xp); yr := yq +(ys – yp)
        линия(xp, yp) |К началу малого квадрата
        Сторона(xp, yp, xq, yq, n - 1, k)
        Сторона(xq, yq, xr, yr, n - 1, k)
        Сторона(xr, yr, xs, ys, n - 1, k)
        линия(xb, yb) |к концу отрезка
   все
кон
```

Основная часть программы:

```
алг Картинка
нач цел n, xc, yc, b, вещ k
    |Включение графического режима работы
    …
    xc := максX/2; yc := максY/2
    b := 100  |Половина длины стороны базового квадрата
    n := 5; k := 0.4
    поз(xc - b, yc - b) |Точка, с которой начинается рисунок
    Сторона(xc - b, yc - b, xc + b, yc - b, n, k)
    Сторона(xc + b, yc - b, xc + b, yc + b, n, k)
    Сторона(xc + b, yc + b, xc - b, yc + b, n, k)
    Сторона(xc - b, yc + b, xc - b, yc - b, n, k)
кон
```

Рекурсия используется при решении известной задачи «Ханойские башни», в которой речь идёт о перекладывании дисков со стержня на стержень по определённым правилам.

Обычно рекурсию образно иллюстрируют примером «У попа была собака, он её любил...» или фантиком конфет «А ну-ка отними!», на котором изображены конфеты «А ну-ка отними!»...

Есть также два стихотворения:

1) *Шёл по улице жучок*
В модном пиджачке.
На груди блестел значок,
А на том значке
Нарисован был жучок,
Тоже в пиджачке.
И на нём висел значок,
А на том значке
Был ещё один жучок…
Автор – Андрей Усачёв

2) *В некий день*
один поэт с мозгами набекрень
поэму сел писать,
начавши:
В некий день
один поэт с мозгами набекрень
поэму сел писать,
начавши:
В некий день …

Если о рекурсии вы всё поняли и «думаете, что это легко, попробуйте проделать это сами» (см. эпиграф к данной статье).

Итак, задания для самостоятельной работы.

1. Составить программу, в результате выполнения которой на экран 10 раз выводится текст истории о попе и его собаке.

Указания по выполнению. Для того чтобы наблюдать вывод каждой новой истории, предусмотрите в процедуре небольшую паузу в выполнении программы.

2. Составить программы для расчёта:

1) значения a^n для заданных значений a и n (n – вещественное число, n – натуральное);

2) n-го члена арифметической прогрессии;

3) то же, геометрической;

4) суммы n первых членов арифметической прогрессии;

5) то же, геометрической;

6) минимального элемента массива;

7) индекса минимального элемента массива.

3. Составить программу, в которой вводятся 10 заданных чисел, а затем они выводятся на экран в обратном порядке. Каждое из заданных чисел должно вводиться один раз. Массивы не применять.

4. Дано натуральное число n. Вывести все его делители в порядке их возрастания. Проверку возможных делителей числа n проводить только до значения, не превышающего $\sqrt{n}$ (все остальные делители получаются в результате деления числа n на найденные делители).

Все задачи решить с использованием рекурсивных функций и процедур.

Дополнение 1

Рекурсия может быть применена для решения задач, в которых имеются так называемые «рекуррентные соотношения» (см. [3]). Если расчет по рекуррентным формулам начинается с известных значений, то при использовании рекурсии эти значения являются «барьером» для прекращения рекурсивных вызовов (которые начинаются от значений, для которых надо определить искомую величину).

Предлагаю читателям решить задачи, представленные в [3], с использованием рекурсии.

1. Начав тренировки, лыжник в первый день пробежал 10 км. Каждый следующий день он увеличивал пробег на 10% от пробега предыдущего дня. Определить, какой суммарный путь он пробежал за первые 7 дней тренировок.

2. В некотором году (назовём его условно первым) на участке в 100 гектар средняя урожайность ячменя составила 20 центнеров с гектара. После этого каждый год площадь участка увеличивалась на 5%, а средняя урожайность на 2%. Определить, какой урожай будет собран за первые шесть лет.

3. Последовательность чисел a_0, a_1, a_2, ... образуется по закону: $a_0 = 1$; $a_k = a_{k-1} + 1/k$ ($k = 1, 2, \ldots$). Дано натуральное число n ($n \geq 2$). Получить a_n.

4. Последовательность чисел v_1, v_2, v_3, ... образуется по закону:

$$v_1 = v_2 = 0;\ v_3 = 1{,}5$$

$$v_i = \frac{i+1}{i^2+1} = v_{i-1} - v_{i-2}v_{i-3}, \quad i = 4, 5, \ldots$$

Дано натуральное число n ($n \geq 4$). Получить v_n.

5. Найти 10-й член последовательности, начинающейся с числа 2,5, в которой каждый следующий член равен сумме обратных величин всех предыдущих.

Дополнение 2

Существует также разновидность рекурсии, которую называют *косвенной*, или *непрямой*. Такой рекурсией является ситуация, когда процедура **А** вызывает себя в качестве вспомогательной не непосредственно, а через другую вспомогательную процедуру **Б**. Такая рекурсия представлена в одном из демонстрационных вариантов ЕГЭ по информатике последних лет:

Ниже на двух языках программирования записаны две рекурсивные процедуры: *F* и *G*.

Алгоритмический язык	Паскаль

алг F(**цел** n)
нач
  если n > 0
  то
  G(n - 1)
  все
кон
алг G(**цел** n)
нач
  вывод "*"
  если n > 1
  то
  F(n - 3)
  все
кон
	procedure F(n: integer);
	begin
	  if n > 0 **then**
	  G(n – 1)
	end;
	procedure G(n: integer);
	begin
	  write('*');
	  if n > 1 **then**
	  F(n – 3)
	end;

Сколько символов «звёздочка» будет напечатано на экране при выполнении вызова *F*(11)?

Образно косвенную рекурсию можно описать так. Перед зеркалом 1 стоит зеркало 2. Что видно в зеркале 1? Зеркало 2, в котором отражается само зеркало 1. В последнем видно зеркало 2 и т. д.

Литература

1. *Грацианова Т.Ю.* Эта страшная рекурсия: рекурсивные процедуры и функции. / Потенциал, 2014, № 5.

2. *Кушниренко А.Г., Лебедев А.Г., Зайдельман Я.Н.* Информатика 7–9: Учебник для общеобразовательных учебных заведений. – М.: Дрофа, 2000.

3. *Златопольский Д.М.* Рекуррентные соотношения. / Потенциал, 2020, № 7.