Проблема метки в программировании

Проблему метки, вернее, проблему избавления от метки, мы рассмотрим на примере решения старинной английской задачи о *рыбаках и рыбке*:

«Три рыбака легли спать, не пересчитав и не поделив улова. Ночью один из рыбаков проснулся и решил уйти, забрав свою долю. Но число рыб не делилось на три. Тогда он, не долго думая, выбросил одну рыбку, а из остатка забрал треть. Второй и третий рыбак поступили аналогичным образом – ушли по-английски и по-джентельменски, выбросив по одной рыбке и оставив спящим товарищам четное число рыб. Спрашивается, какое минимальное число рыб в улове отвечает условию задачи».

Пусть читатель сначала попробует решить эту задачу без компьютера. Компьютер же мы привлечем к этой работе для двух целей. Во-первых, на задаче о рыбаках и рыбке мы рассмотрим некоторые методы *структурирования* программ в среде Mathcad – методы освобождения программ от меток и «втискивания» их в узкие рамки структурных управляющих конструкций, описанных выше (см. рис. 6.2). Во-вторых, мы покажем, что задача о рыбаках и рыбке до сих пор решалась неправильно...

Эту задачу в среде Mathcad можно решить, не прибегая к средствам программирования.

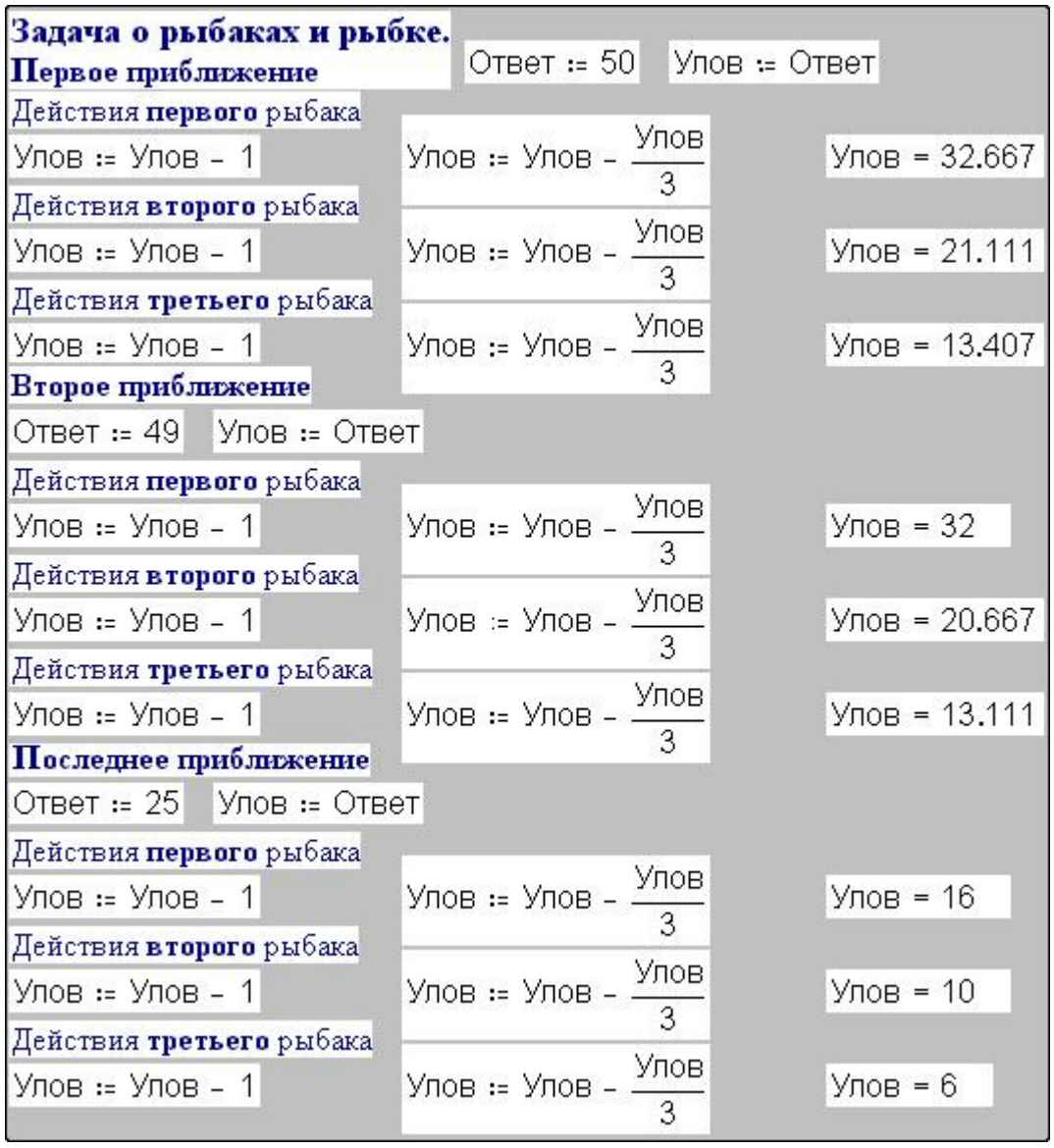

Рис. 1. Задача о рыбаках и рыбке: «беспрограммное» решение в среде Mathcad

На рис. 1 показано, как задача решается методом последовательных приближений – задается первое приближение к ответу (50 рыб), а затем от этого числа отнимается по единице до тех пор, пока убывающий улов не будет представлять собой целочисленный ряд: было 25 рыб (искомый ответ задачи), первый рыбак выбросил одну, забрал треть и оставил товарищам 16 рыб (по 8 каждому); второй рыбак (не зная, что первый ушел) оставил 10 рыб, а третий – 6. Задача решена, но с применением «ручной» работы, состоящей в наблюдении за значениями переменной Улов и в изменении (уменьшении на единицу) значения переменной Ответ. (Блоки операторов, фиксирующих действие трех рыбаков, можно не дублировать, как

это сделано на рис. 1. Достаточно уменьшать[1] значение переменной Ответ и следить за значениями переменной Улов).

Попробуем автоматизировать поиск ответа в задаче о рыбаках и рыбке.

```
‘ 1. Исходная неструктурированная Basic-программа
Input "Предположение"; Ответ
label: Улов = Ответ
        For Рыбак = 1 To 3
                Улов = Улов – 1
                Улов = Улов - Улов / 3
                If Улов > Int(Улов) Then Ответ = Ответ - 1: Goto label
        Next
Print "Ответ "; Ответ; “рыб”


‘ 2. Первый шаг структурирования - разбег
Input "Предположение"; Ответ
Ответ = Ответ + 1 ‘ Шаг назад
label: Ответ = Ответ - 1 ‘ Шаг вперед
        Улов = Ответ
        For Рыбак = 1 To 3
                Улов = Улов – 1
                Улов = Улов - Улов / 3
                If Улов > Int(Улов) Goto label
        Next
Print "Ответ "; Ответ; “рыб”
```

[1] Метод последовательных приближений широко используется в инженерных расчетах. Если блок операторов, по которым рассчитывается новое приближение, объемный, то можно рекомендовать «опустить» оператор задания первого (предыдущего) приближения к оператору вывода значения очередного приближения, записав там Ответ≡50 (если говорить о программе на рис. 5.22). Это исключит «ползанье» по Mathcad-документу.

```
‘ 3. Второй шаг структурирования – ввод признака
Input "Предположение"; Ответ
Ответ = Ответ + 1
label: Ответ = Ответ – 1
        Улов = Ответ
        Поделили = "да" ’ Признак дележа улова
        For Рыбак = 1 To 3
                Улов = Улов – 1
                Улов = Улов - Улов / 3
                If Улов > Int(Улов) Then Поделили = “нет”
        Next
If Поделили = “нет Goto label
Print "Ответ "; Ответ; “рыб”

‘ 4. Третий шаг структурирования – отказ от метки
Input "Предположение"; Ответ
Ответ = Ответ + 1
Do ’ Начало цикла с постпроверкой
        Ответ = Ответ – 1
        Улов = Ответ
        Поделили = "да"
        For Рыбак = 1 To 3
                Улов = Улов – 1
                Улов = Улов - Улов / 3
                If Улов > Int(Улов) Then Поделили = “нет”
        Next
Loop Until Поделили = "да" ’ Конец цикла
```

```
Print "Ответ "; Ответ; “рыб”
```

Рис. 2. BASIC-программы решения задачи о рыбаках и рыбке

На рис. 2 представлены четыре Basic-программы, решающие задачу о рыбаках и рыбке в автоматическом режиме: оператор Input запрашивает первое приближение (те же 50 рыб, к примеру), а оператор Print выдает ответ – 25 рыб. В первой BASIC-программе один к одному реализован алгоритм «ручного» расчета: как только в теле цикла с параметром (три последовательных ухода рыбаков) переменная Улов обретает дробный «хвостик» (встроенная BASIC-функция Int этот «хвостик» обрезает; ее Mathcad-аналог – функция floor), то (Then) предположение уменьшается на единицу (Ответ = Ответ – 1), а управление программой передается к оператору, помеченному меткой (**Goto** label). Прежде чем эту программу перевести на язык Mathcad, ее нужно освободить от метки[2]. И не только потому, что в арсенале средств программирования Mathcad нет метки и операторов условного и безусловного перехода к метке, но по другим причинам, не связанным с Mathcad. В нашей коротенькой программе-безделушке (пункт 1 на рис. 2) метка вполне уместна и естественна, но если программа с метками разрастается, то в ней становится трудно разбираться и ее практически невозможно отлаживать и расширять. Программа, как справедливо подчеркивают адепты структурного программирования, становится похожа на спагетти: вытаскиваешь (вилкой) блок операторов для отдельной отладки или компиляции, а к нему намертво привязаны нити (макаронины) Goto-переходов. Кроме того, такую программу невозможно создавать группой разработчиков (технология снизу вверх). Первые реализации языка Pascal совсем не имели меток, так как этот язык разрабатывался Н.Виртом в первую очередь для обучения студентов структурному программированию. Метка появилась только

[2] Структурная революция началась со статьи Э. Дейкстры «Programming without GOTO». Сейчас многие программисты-снобы, как нами уже отмечено, хвастают тем, что они написали не одну сотню программ, ни разу не обратившись к метке.

в поздних версиях этого языка. Так от детей в период, когда они учатся жить (выживать!), прячут спички.

Первый шаг структурирования BASIC-программы на рис. 2 – это превращение конструкции:

```
If Улов > Int(Улов) Then Ответ = Ответ - 1: Goto label
```

в оператор условного перехода к метке:

```
If Улов > Int(Улов) Goto label
```

Это сделать несложно (см. пункт 2 на рис. 2), применив в программе технику разбега спортсмена перед прыжком: шаг назад от черты-метки (Ответ = Ответ + 1) и разбег (Ответ= Ответ - 1). Структурирование программы, как правило, несколько усложняет алгоритм: «За все нужно платить!», «Красота требует жертв!» и т. д.

Второй шаг структурирования – это вытаскивание оператора безусловного перехода из тела цикла For. Для этого в программу (см. пункт 3) вводится вспомогательная булева переменная-признак Поделили, а в теле цикла оператор условного перехода к метке заменяется на оператор «альтернатива с одним плечом» (одна из основных структурных управляющих конструкций). Сам же оператор условного перехода «сползает» вниз. После этого в программе «вырисовывается» еще одна основная структурная управляющая конструкция – цикл с постпроверкой, который в третьем варианте на рис. 2 реализован через метку и оператор условного перехода к метке. После этого программу наконец-то можно совсем освободить метки, реализуя алгоритм через цикл с постпроверкой, в тело которого вписан цикл for, а в него — альтернатива с одним плечом (пункт 4).

После всех этих манипуляций четвертый вариант BASIC-программы можно один к одному переписать для Mathcad – см. рис. 3. Придется только оформить ее в виде функции

пользователя: в языке Mathcad нет операторов Input и Print[3]. Их аналоги в среде Mathcad (операторы ■ := ■ и ■ =) работают, увы, только вне программ.

```
Ответ( Ответ) :=  "Задача о рыбаках и рыбке – цикл с постпроверкой"
                  Ответ ← Ответ + 1
                  while 1
                      Ответ ← Ответ - 1
                      Улов ← Ответ
                      Поделили ← "да"
                      for Рыбак ∈ 1, 2, 3
                          Улов ← Улов - 1
                          Улов ← Улов - Улов / 3
                          Поделили ← "нет" if Улов > floor( Улов )
                      break if Поделили = "да"
                  Ответ

"Нормальный" ответ  Ответ( 50) = 25
Ответ Дирака        Ответ( 24) = -2
Наш ответ Дираку    Ответ( -3) = -29
                    Ответ( -30) = -56   и т.д.

Правильный ответ – бесконечный ряд чисел ("рыбый ряд") с периодом
Ответ( 26) - Ответ( 24) = 27
```

Рис. 3. Mathcad-программа решения задачи о рыбаках и рыбке

На рис. 3 показаны вызовы функции Ответ при различных значениях предположений (50, 24, минус 3 и даже минус 30 рыб). Английский физик Поль Дирак придумал не только античастицы, но и «антирыбы»: он сказал, что задача о рыбаках и рыбке решалась неправильно (25 рыб). Правильный ответ – минус две рыбы (плюс две антирыбы): выбрасываем одну – остается минус три, забираем треть – остается минус две и так до бесконечности. Наше компьютерное решение задачи показывает, что и Дирак ошибался:

[3] В языке Visual Basic оператора Print тоже нет, а есть *метод* Print.

«Поль, ты не прав!» Условию задачи отвечает бесконечный ряд чисел (назовем его «рыбный ряд Дирака») с шагом 27.

Чтобы не прослыть совсем уж полным педантом (программистом-занудой), можно в конец цикла for на рис. 3 вставить оператор break if Поделили = ”нет”, прерывающим выполнения цикла for. Если рыбаков будет не трое, а больше (тридцать три рыбака, например – задача для читателя), то этот прием существенно ускорит работу программы (см. главку «Оптимизация Mathcad-программ»).

Можно еще усложнить задачу, заставив любое количество рыбаков выбрасывать или подлавливать любое количество рыб.

Задачу о рыбаках и рыбке можно решать другим способом – перебором с другого конца: задать не начальное число рыб в улове, а предположить, что последний рыбак оставляет две рыбы (меньше не может быть), и увеличивать их число на единицу, если условия задачи не выполняются. Задача будет решаться быстрее, но минус двух рыб, а, тем более, минус 29 рыб мы не получим: «Keep if simple, stupid! – Делай это проще, дурачок!» Человеку психологически трудно спуститься к отрицательным числам в ответе, машина же делает это спокойно, без всяких предрассудков. Не дает отрицательного ответа и аналитическое решение задачи – поиск целочисленных корней одного уравнения с двумя неизвестными.