«Мир информатики»
Журнал для тех, у кого информатика – любимый школьный предмет

Выпуск № 36, август 2019 г.

01100010001010100100111110010001100100100010100100100101 1

Школа программирования

Об использовании процедур и функций

Что такое «процедура» и что такое «функция»? Об обоих[1] этих понятиях можно сказать, что это – фрагмент программы, в котором решается какая-то часть всей задачи. Эта фрагмент оформляется особым образом и снабжается именем. Возникает вопрос: «Зачем нужно какую-то часть программы оформлять отдельно?». Чтобы ответить на него, рассмотрим несколько примеров.

В программах часто встречаются повторяющиеся или похожие фрагменты. Например, для того, чтобы вывести на экран «заставку» с изображением, представленным на рис. 1,

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        ЭТУ ПРОГРАММУ РАЗРАБОТАЛ ПЕТЯ ХАКЕРОВ
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Рис. 1

– в программе должно быть записано:

```
нц для i от 1 до 40
  вывод "* "
кц
вывод нс
нц для i от 1 до 10
  вывод " "
кц
вывод нс, "ЭТУ ПРОГРАММУ РАЗРАБОТАЛ МИТЯ ХАКЕРОВ"
вывод нс
нц для i от 1 до 40
  вывод "* "
кц
```

Видно, что в программе есть два абсолютно одинаковых фрагмента, относящиеся к выводу линий из звездочек. Можно эти фрагменты оформить как процедуру.

[1] В языках программирования Си, Python и некоторых других оба эти понятия называются общим термином «функция».

На школьном алгоритмическом языке общий вид процедуры следующий[2]:

```
алг <имя_процедуры>
нач <описание_переменных_величин_используемых_в_процедуре>
  <тело_процедуры>
кон
```

где *<описание_переменных_величин_используемых_в_процедуре>* – перечень используемых в процедуре величин с указанием их типа; *<тело_процедуры>* – операторы, реализующую решаемую с помощью процедуры задачу.

В нашей задаче процедура, с помощью которой можно получить линию из 40 звездочек, имеет вид:

```
алг Линия
нач цел i
  нц для i от 1 до 40
    вывод "* "
  кц
кон
```

Созданную процедуру можно использовать в основной части программы. Вызов процедуры на выполнение осуществляется отдельным оператором, в котором указывается имя процедуры:

```
Линия
вывод нс
нц для i от 1 до 10
  вывод " "
кц
вывод нс, "ЭТУ ПРОГРАММУ РАЗРАБОТАЛ МИТЯ ХАКЕРОВ"
вывод нс
Линия
```

Нетрудно убедиться, что данном случае применение процедуры уменьшает размер программы (это преимущество проявилось бы в еще большей степени, если бы в задаче пришлось рисовать линии 3 и более раз).

Еще один пример – пусть в значительной степени условный, но достаточно показательный. Если необходимо на экране нарисовать из линий изображение слова «МУ», то соответствующая программа оформляется так:

```
алг МУ
нач
  …
  сместиться на вектор(0, 100)
  сместиться на вектор(50, –50)
  сместиться на вектор(50, 50)
  сместиться на вектор(0, –100)
  поднять перо
  сместиться на вектор(10, 0)
  опустить перо
  сместиться на вектор(50, 0)
  сместиться на вектор(0, 100)
  …
кон
```

[2] С правилами оформления процедур и функций в языке программирования, которым вы владеете, ознакомьтесь самостоятельно.

Легко ли в такой программе найти ошибочный оператор, из-за которого, например, буква У изображена неправильно? Где нужно добавить операторы, чтобы получить на экране вместо «МУ» – слово «МЯУ»?

Если же рисование букв М и У оформить в виде отдельных процедур с соответствующими именами, то основная часть программы примет вид:

```
алг МУ
нач
  сместиться в точку(50, 100)
  М | Вызов процедуры М
  сместиться в точку(160, 100)
  У | Вызов процедуры У
кон
```

Конечно, при этом размер всей программы несколько увеличится, но зато проявятся другие преимущества – программа станет более понятной, в ней легко найти нужное место, в том числе и связанное с ошибками. Если же нужно изобразить на экране слово МУМУ, то тогда применение процедур, кроме того, приведет и к уменьшению размера программы.

Возможность использования процедур и функций позволяет применять современные методы проектирования программ. Дело в том, что при программировании достаточно сложной задачи решить ее «одним махом», т. е. написать последовательно всю программу от начала до конца, обычно невозможно. Процесс разработки программ – это творческий процесс, проходящий в несколько этапов. Вначале стараются разработать наиболее общую схему алгоритма, не останавливаясь на технических деталях его реализации. В результате такой алгоритм представляется в виде последовательности относительно крупных блоков, реализующих более или менее самостоятельные смысловые части алгоритма, в которых решаются какие-то частные задачи. Эти блоки, в свою очередь, могут разбиваться на меньшие подблоки и т. д. Процесс последовательного структурирования программы продолжается до тех пор, пока реализуемые блоками алгоритмы не станут простыми и легко программируемыми.

Рассказав о преимуществах применения процедур, пойдем дальше.

Возможно также использование так называемых «процедур с параметрами» – процедур, результат выполнения которых зависит от некоторых величин – их параметров. Пусть, например, в программе требуется рисовать линии из звездочек разной «длины»:

```
…
нц для i от 1 до 40
  вывод "* "
кц
…
нц для i от 1 до 20
  вывод "* "
кц
…
нц для i от 1 до 30
  вывод "* "
кц
```

Можно, конечно, для решения такой задачи создать три отдельные процедуры, решающие эти задачи. Но выиграем ли мы при этом в размере программы? Нет, конечно. Желательно разработать процедуру, которая «умеет» рисовать линии из звездочек любой длины. Это и будет упомянутая чуть выше процедура с параметром.

Общий вид таких процедур:

```
алг <имя_процедуры> (<список_формальных_параметров>)
нач <описание_переменных_величин_используемых_в_процедуре>
  <тело_процедуры>
кон
```

где *<список_формальных_параметров>* – перечень величин, от которых зависит результат выполнения процедуры (с указанием их типа); эти величины называют «формальными параметрами процедуры». Формальные параметры используются в теле процедуры.

В нашем случае процедура, с помощью которой можно получить линию любой «длины», имеет вид:

```
алг Линия(цел k)
нач цел i
  нц для i от 1 до k
    вывод "*"
  кц
кон
```

Результат ее выполнения зависит от величины `k`, которая и является единственным параметром процедуры.

Вызов процедуры с параметрами осуществляется так:

```
<имя_процедуры> (<список_фактических_параметров>)
```

где *<список_фактических_параметров>* – перечень конкретных значений параметров процедуры для решения конкретной задачи. Так, в нашем случае, для того чтобы использовать процедуру для рисования линии из 30 звездочек, вызов процедуры должен быть оформлен следующим образом:

```
Линия(30)
```

В качестве значений фактических параметров можно указывать[3]:

1) константы (заданные, известные значения) – именно так было сделано в приведенном примере;

2) имена величин, например, `Линия(длина1)`; как видно из примера, имена фактических и формальных параметров не обязательно должны совпадать;

3) выражения (`Линия(длина1 + 10)`).

В каждом из этих трех случаев тип фактического параметра должен совпадать с типом формального параметра, указанным в заголовке процедуры.

Ясно, что в процедуре могут быть использованы два и более формальных параметра. В таких случаях при оформлении списка фактических параметров следует учитывать следующие требования:

1) количество фактических параметров должно быть таким же, как и в списке формальных параметров в описании процедуры;

2) тип каждого фактического параметра должен совпадать с типом соответствующего формального параметра;

3) соответствующие фактические и формальные параметра должны совпадать по смыслу.

[3] Строго говоря, это не всегда так, но в данной статье мы не будем описывать другие варианты.

Если первые два требования ясны и понятны, то третье требует комментариев. Пусть, например, подготовлена процедура, изображающая на экране прямоугольник:

```
алг Линия(цел a, b)
нач
  …
кон
```

Как с ее помощью нарисовать прямоугольник шириной 20 и высотой 100? Какой из вариантов оформления вызова процедуры является правильным: `Линия(20, 100)` или `Линия(100, 20)`? Ответ зависит от того, какому размеру соответствует формальный параметр *a*, а какому – *b*. Если *a* – это высота прямоугольника, то правильный вариант – `Линия(100, 20)`, в противном случае первой должна быть указана ширина изображаемого прямоугольника, а второй – высота (`Линия(20, 100)`).

Теперь о том, что такое «функция».

Вы, конечно, знаете о так называемых «стандартных» (имеющихся в языке программирования) функциях – *sin, sqrt* и др. Что они делают при их использовании в программах? – Возвращают какое-то значение (результат расчетов или т. п.). Так вот, с такой же целью можно также создавать и использовать собственные функции. Пусть, например, в программе надо рассчитать значения *x*:

$$x = \frac{2+\sqrt{2}}{5+\sqrt{5}} + \frac{5+\sqrt{5}}{13+\sqrt{13}} + \frac{11+\sqrt{11}}{8+\sqrt{8}} + \frac{14+\sqrt{14}}{7+\sqrt{7}}$$

Видно, что в выражении имеются похожие фрагменты – дроби. Желательно для расчета дробей оформить функцию, которая вычисляла бы значения отдельных дробей вида $\frac{a+\sqrt{a}}{b+\sqrt{b}}$ при любых значениях *a* и *b*, а затем использовать ее для расчетов.

Правила оформления функций в разных языках программирования различаются. В школьном алгоритмическом языке общий вид функций такой:

```
алг <тип_результата> <имя_функции> (<список_формальных_параметров>)
нач <описания_переменных_величин_используемых_в_функции>
  <тело_функции>
кон
```

причем последним оператором тела функции должен быть оператор присваивания, в левой части которого должно быть указано служебное слово **знач**.

В нашем случае функция для расчета значения дробей указанного чуть выше вида оформляется так:

```
алг вещ Др(арг цел a, b)
нач
  знач := (a + sqrt(a))/(b + sqrt(b))
кон
```

Как и стандартные, функция, созданная программистом, используется в правой части оператора присваивания. Для этого надо указать ее имя, а в скобках – значения фактических параметров:

```
x := Др(2, 5) + Др(5, 13) + Др(13, 8) + Др(14, 7)
```

Видно, что благодаря использованию функции, оператор для расчета значения *x* гораздо компактнее такого варианта:

```
x := (2 + sqrt(2))/(5 + sqrt(5)) + (5 + sqrt(5))/(13 + sqrt(13)) +
(13 + sqrt(13))/(8 + sqrt(8)) + (14 + sqrt(14))/(7 + sqrt(7))
```

Кроме того, в последнем варианте выше вероятность появления ошибки при записи оператора.

Еще одно преимущество использования собственных функций – улучшение читаемости (понятности) программы. Например, задачу «Даны 20 целых чисел. Определить, сколько из них имеют ровно 4 делителя» можно решить так:

```
алг Задача_о_числах_с_четырьмя_делителями
нач цел n, возм_дел, кол_дел, кол_4дел, номер
   кол_4дел := 0  | Начальное значение искомой величины
   нц для номер от 1 до 20
    ввод n
    кол_дел := 0  | Начальное значение количества делителей числа n
    нц для возм_дел от 1 до n | Рассматриваем все возможные делители
     если mod(n, возм_дел) = 0
      то | Число возм_дел – делитель числа n
         кол_дел := кол_дел + 1
     все
    кц
    если кол_дел = 4
     то
      кол_4дел := кол_4дел + 1
    все
   кц
   вывод кол_4дел
кон
```

Если же разработать функцию, определяющую количество делителей числа n:

```
алг цел Кол_дел(цел n)
нач цел возм_дел, кол_дел
   кол_дел := 0
   нц для возм_дел от 1 до n
    если mod(n, возм_дел) = 0
     то
      кол_дел := кол_дел + 1
    все
   кц
   знач := кол_дел
кон
```

и использовать ее в программе решения «главной» задачи:

```
алг Задача_о_числах_с_четырьмя_делителями
нач цел n, кол_4дел, номер
   кол_4дел := 0
   нц для номер от 1 до 20
    ввод n
    если Кол_дел(n) = 4
     то
      кол_4дел := кол_4дел + 1
    все
   кц
   вывод кол_4дел
кон
```

то такая программа, безусловно, будет понятнее, чем приведенная первой.

Результат, возвращаемый функцией, может быть любого типа, в том числе логического[4]. Например, функция, определяющая, является ли натуральное число *a* делителем натурального числа *b*, имеет вид:

```
алг лог Делитель(цел a, b)
нач цел
  |Значение функции:
  знач := mod(b, a) = 0
кон
```

Такая функция может быть использована в другой функции – в возвращающей количество делителей числа (см. чуть выше):

```
алг цел Кол_дел(цел n)
нач цел возм_дел, кол_дел
  кол_дел := 0
  нц для возм_дел от 1 до n
    если Делитель(n, возм_дел)
      то
        кол_дел := кол_дел + 1
    все
  кц
  знач := кол_дел
кон
```

Важно понимать, что происходит в памяти компьютера при использовании процедур (или функций). Когда программа запускается на выполнение, она размещается в оперативной памяти. Это можно смоделировать в виде прямоугольника; в этом прямоугольнике записаны операторы программы:

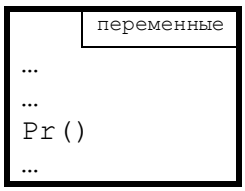


Рис. 2

Предусмотрено также место для размещения значений всех переменных программы.

Операторы программы начинают выполняться. Когда доходит очередь до оператора вызова процедуры `Pr()`, то:

1) выполнение основной части программы приостанавливается;

2) в памяти отводится место для размещения данной процедуры. Это также можно смоделировать в виде прямоугольника с текстом процедуры:

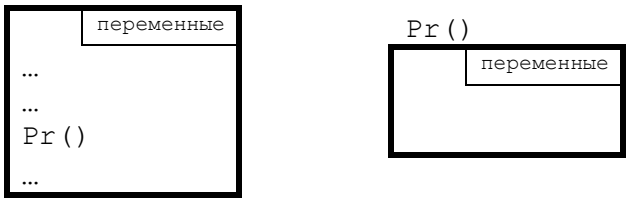


Рис. 3

В «новом» прямоугольнике также предусмотрено место для размещения значений всех переменных, использованных в процедуре;

[4] Если, конечно, такой тип предусмотрен в языке программирования.

3) выполняются операторы процедуры;

4) после выполнения всех операторов процедуры место в памяти для нее освобождается:

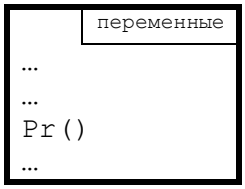


Рис.4

После этого продолжается выполнение операторов основной части программы. Если в программе еще раз встречается вызов процедуры – происходит то же самое.

Заметим, что переменные величины, описанные в процедуре или функции, называют «локальными», а описанные в основной части программы – «глобальными»[5].

Из приведенного порядка работы программы при использовании процедур и функций следует, что локальные переменные «живут» только во время выполнения функции и поэтому их значения недоступны в основной части программы и в других процедурах или функциях. Глобальные же переменные «внутри» процедур и функций можно использовать.

Например, при выполнении следующих двух программ появится сообщение об ошибке, связанное с этим обстоятельством:

1.

```
алг Основная_часть_программы[6]
нач
  Процедура1
  вывод a
кон
алг Процедура1
нач цел a
  a := 100
кон
```

2.

```
алг Основная_часть_программы
нач
  Процедура2
  Процедура1
кон
алг Процедура2
нач цел a
  a := 100
кон
алг Процедура1
нач цел a
  вывод a
кон
```

[5] В школьном алгоритмическом языке глобальные переменные объявляются (описываются) до основной части программы, в языке программирования Python описание глобальных переменных проводится по-особому.

[6] В школьном алгоритмическом языке основная часть программы размещается раньше всех вспомогательных процедур и функций.

В то же время глобальные переменные доступны («видны») в процедурах:

```
цел a | Глобальная переменная
алг Основная_часть_программы
нач
  a := 100
  Процедура1
кон
алг Процедура1
нач
  вывод a
кон
```

Еще один случай, который следует рассмотреть, – совпадение имен локальной и глобальной переменных. Как, по-вашему, что будет выведено на экран в результате выполнения следующей программы:

```
цел a
алг Основная_часть_программы
нач
  a := 100
  Процедура1
кон
алг Процедура1
нач цел a
  a := 1
  вывод a
кон
```

Правильный ответ – 1 (говорят, что в этом случае локальная переменная «закрывает» собой глобальную).

Предлагаем читателям обратить внимание на эти важные обстоятельства.

В заключение заметим, что функция/процедура может в процессе своей работы использовать любую другую функцию/процедуру (это было продемонстрировано выше) или саму себя. В последнем случае такие вспомогательные программы называется «рекурсивными». Рекурсивным функциям и процедурам посвящена следующая статья.

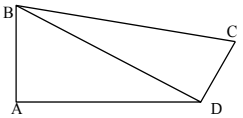
Задачи для самостоятельной работы

1. Рассчитать значение *x*, определив и использовав функцию:

$$x = \frac{\sqrt{6}+6}{2} + \frac{\sqrt{13}+13}{2} + \frac{\sqrt{21}+21}{2}$$

2. Найти периметр треугольника, заданного координатами своих вершин. (Определить функцию для расчета длины отрезка по координатам его вершин.)

3. Найти периметр фигуры ABCD по заданным сторонам AB, AD и CD. (Определить функцию для расчета гипотенузы прямоугольного треугольника по его катетам.)



$\angle BAD=90^0$, $\angle BDC=90^0$

4. Определить количество шестизначных счастливых чисел. «Счастливым» называют такое шестизначное число, что сумма его первых трех цифр равна сумме его последних трех цифр. (Определить функцию для расчета суммы цифр трехзначного числа).

5. Даны два предложения. В каком из них доля (в %) буквы «д» больше. (Определить функцию для расчета доли некоторой буквы в предложении.)

6. Даны 20 чисел. Определить, сколько из них являются простыми. Использовать функцию, распознающую простые числа.

7. Дана последовательность чисел 2, 3, 5, 9, 17, …, задаваемая следующим рекуррентным соотношением[7] $a_i = 2 \times a_{i-1} - 1$ ($a_1 = 2$). Определить значение k-го члена указанной последовательности ($1 \le k \le 50$). Для этого разработать функцию, в которой используется и заполняется массив из 50 элементов, и которая возвращает значение k-го элемента массива.

Рекурсия. Рекурсивные функции и процедуры

«Рекурсивными»[8] называют функции и процедуры[9], которые используют (вызывают) как вспомогательную саму себя.

В качестве примера приведем функцию для расчета 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(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, как указывалось в *предыдущей статье*, при каждом вызове подпрограммы для нее отводится место в оперативной памяти, которое «освобождается» после завершения ее (функции/процедуры) работы. Но если во вспомогательной функции/процедуре имеется рекурсия, то вызовы 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 приведенной функции это условие записывается так:

```
алг цел Факториал(цел n)
нач
  если n > 1
    то
      | Рекурсивный вызов функции Факториал
      знач := Факториал(n - 1) * n
    иначе | Значение функции известно (рекурсивного вызова нет)
```

---

[7] Рекуррентное соотношения – формула, выражающая значение очередного члена последовательности через значение одного или нескольких предыдущих членов и, возможно, номер члена последовательности.

[8] От латинского *recursio* – возвращение.

[9] Как указывакися в предыдушей статье, в языках программирования Си, Python и некоторых других оба эти понятия называются общим термином «функция». В дальнейшем мы будем использовать понятие «подпрограмма», объединяющее оба указанных вида вспомогательных программ.

```
        знач := 1
    все
кон
```

или

```
алг цел Факториал(цел n)
нач
  если n > 1
    то
      знач := 1
    иначе
      знач := Факториал(n - 1) * n
  все
кон
```

Алгоритм вычисления значения функции $F(n)$, использующий рекурсию, может быть описан по-другому – с указанием двух или более вариантов значения, например, в виде:

```
F(1) = 10
F(n) = F(n – 1) + 3 при n > 1
```

или

```
F(1) = 10
F(n) = F(n – 1) + 3 при n > 1
F(n) = 0 при n < 1
```

или

```
F(12) = 5
F(n) = F(n + 1) – 1 при n < 12
```

или

```
F(n) = 25 при n > 12
F(12) = 5
F(n) = F(n + 1) – 1 при n < 12
```

и т. п.

Видно, что в двух первых случаях рекурсивные вызовы осуществляются с меньшими значениями аргумента (фактического параметра – см. *первую статью выпуска*), в остальных – с большими. Во всех случаях когда-то рекурсивные вызовы прекратятся (убедитесь в этом!) и начнется передача результатов в «вызывающую» функцию.

Пример работы программы при вычислении значения факториала числа 4 показан на рис. 1–2.

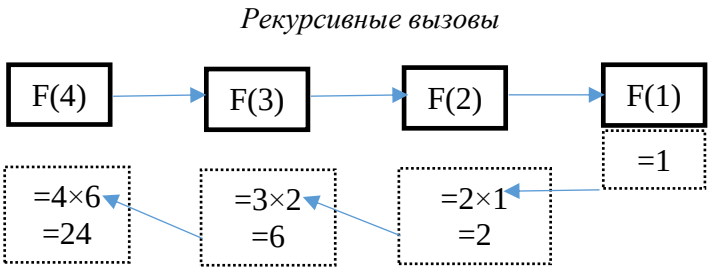


Рис. 1

F(n)	F(4) →	F(3) →	F(2) →	F(1)
Значение функции	24 ← 4 × 6	6 ← 3 × 2	2 ← 2 × 1	1
n	4	3	2	1

Рис. 2

Ответ: 24.

Рассмотрим очень показательную задачу: «Определить 50-й член последовательности Фибоначчи (последовательность Фибоначчи образуют числа 1, 1, 2, 3, 5, 8, 13, ...)».

Достаточно очевидный метод решения задачи такой – использовать массив из 50 элементов, записать в него значения двух первых элементов (равных 1), а затем последовательно вычислять остальные как сумму двух предшествующих значений:

```
алг Расчет
нач цел таб мас[1 : 50], цел k
  мас[1] := 1
  мас[2] := 1
  нц для k от 3 до 50
    мас[i] := мас[i - 2] + мас[i - 1]
  кц
  вывод мас[50]
кон
```

Недостатком такого метода является то, что, хотя нас интересует лишь один член последовательности, в процессе выполнения программы компьютер вычисляет и хранит в памяти все члены последовательности от 1-го до 50-го.

Этого недостатка лишен второй метод вычислений – без использования массивов. Действительно, если для вычисления очередного члена последовательности Фибоначчи нужно знать только значения двух предыдущих, то можно не запоминать все члены последовательности в массиве, а хранить две величины:

пред – элемент, предшествующий очередному члену последовательности;

предпред – элемент, предшествующий элементу *пред*.

Кроме них, используем в программе величину *очер* – очередной рассчитываемый элемент последовательности.

Сначала имеем:

```
пред := 1
предпред := 1
```

Затем рассчитываем очередной (третий) элемент:

```
след := пред + предпред,
```

после чего «готовимся» к расчету следующего элемента:

```
предпред := пред | Именно в
пред := след     | таком порядке!
```

и определяем его:

```
очер := пред + предпред
```

и т. д.

Ясно, что для определения 50-го члена последовательности необходимо провести 48 повторений описанных действий. Итак, программа:

```
алг Расчет_2
нач очер, пред, предпред, k
  пред := 1
  пред пред := 1
  нц для k от 1 до 48
    очер := пред + предпред
    предпред := пред
    пред := очер
  кц
  вывод очер
кон
```

Для решения обсуждаемой задачи может быть использована рекурсия. Вот как просто и логично выглядит рекурсивный вариант функции:

```
алг цел Фиб(цел k)
нач
  если k = 1 или k = 2
    то
      знач := 1
    иначе
      знач := Фиб(k - 2) + Фиб(k - 1)
  все
кон
```

Такое оформление функции полностью соответствует закону построения последовательности Фибоначчи – очередной элемент последовательности равен сумме двух предыдущих. При нем не требуется применять оператор цикла и думать[10] над последовательностью расчета значений *пред* и *предпред*. Есть только одно «но»! (☺).

В процессе вычислений, например, Фиб(17), функция для расчета Фиб(15) будет вызываться 2 раза, для расчета Фиб(14) – 3 раза, Фиб(13) – 5 раз, Фиб(12) – 8 раз и т. д. Всего при вычислении Фиб(17) компьютер выполнит более 1000 операций сложения, Фиб(31) – более миллиона, Фиб(45) – более миллиарда операций сложения!

А при использовании массива (как в программе `Расчет`) или как в программе `Расчет_2` (см. начало статьи) количество сложений в указанных задачах соответственно равно 15, 29 и 43!

Поэтому говорят, что «рекурсия – это эффектно, но не всегда эффективно».

[10] Хотя думать надо всегда! ☺

Фракталы

Вы, возможно, слышали слово «фрактал». Так называют объект, в точности или приближённо совпадающий с частью себя самого, то есть целое имеет ту же форму, что и одна или более частей (от латинского *fractus* — дроблёный, сломанный, разбитый).

Существует простая процедура получения фрактальных кривых на плоскости.

Зададим произвольную ломаную с конечным числом звеньев, называемую «генератором» (например, такую, как верхнее изображение на рис. 1).

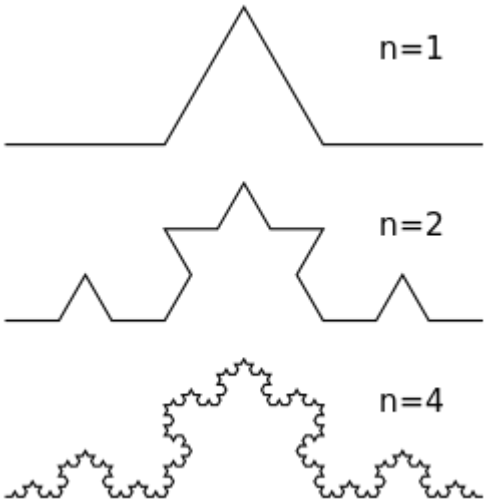


Рис. 1

Далее заменим в ней каждый отрезок генератором (точнее, ломаной, подобной генератору). В получившейся ломаной вновь заменим каждый отрезок генератором. Продолжая до бесконечности, в пределе получим фрактальную кривую. На рис 1 справа приведены номера шагов этой процедуры и соответствующие изображения для получения фрактала — так называемой «кривой Коха» («снежинки Коха»).

На основе описанного алгоритма разрабатываются программы для изображения фракталов на экране компьютера. Как правило, они используют рекурсию (см. рубрику «Школа программирования»).

В 1915 году польский математик и педагог Вацлав Серпинский описал фрактал, который называют «треугольник Серпинского). Чтобы его получить, нужно взять (равносторонний) треугольник, провести в нём средние линии и «выкинуть» центральный из четырех образовавшихся маленьких треугольников. Дальше эти же действия нужно повторить с каждым из оставшихся трех треугольников и т.д. На рис. 2 показаны первые три шага.

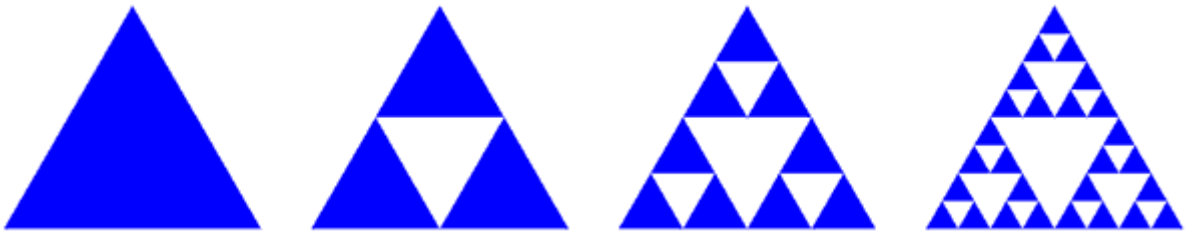


Рис. 2

Еще один фрактал показан на рис. 3. Это так называемый «ковер Серпинского».

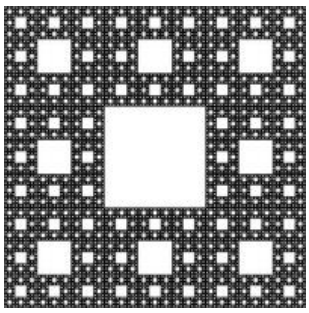


Рис. 3

Эта фигура получается следующим образом.

Квадрат (рис. 4) разобьем на 9 одинаковых квадратов (рис. 5), после чего средний квадрат оттеним.

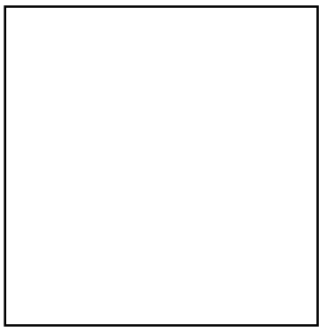


Рис. 4

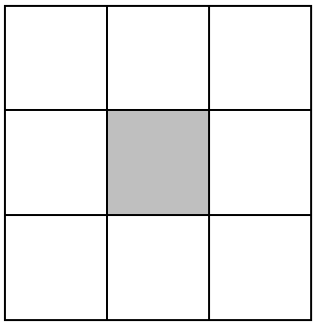


Рис. 5

Затем каждый из новых квадратов, кроме оттененного, также разобьем на 9 одинаковых квадратов и также оттеним средние квадраты (рис. 6).

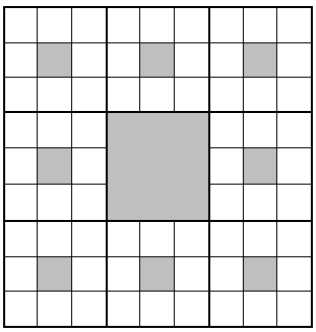


Рис. 6

После 4-го шага получим изображение, аналогичное показанному на рис. 1 (отличие будет в том, что мы оттеняли квадраты, которые на рис. 1 – белые.

Сколько неоттененных квадратов минимального размера будет после 8-го «шага»? Сколько всего оттененных разного размера?

Ответы присылайте в редакцию.

Ответы, решения, разъяснения

к заданиям, опубликованным в журнале «Мир информатики» № 33 (апрель 2019 г.)

Ребусы по информатике

Ответы:

Ребус № 1. Задача	Ребус № 5. Разработка
Ребус № 2. Алгоритм	Ребус № 6. Ошибка
Ребус № 3. Программа	Ребус № 7. Отладка
Ребус № 4. Программист	Ребус № 8. Робот

Правильные ответы прислали:

— Андрющенко Анастасия, Гольтяпина Ева, Орлова Арианна и Самсоненко Юлия, школа № 1 г. Зеленокумска Ставропольского края, учитель **Букина Е.Ю.**;

— Авакиян Евгений, Медведев Владимир, Попкова Мария, Резцова Анастасия и Ягиева Гуля, средняя школа села Ириновка, Новобурасский р-н Саратовской обл., учитель **Брунов А.С.**;

— Востриков Максим и Чуркин Владислав, г. Пенза, школа № 73, учитель **Диков А.В.**;

— Невоструев Семён, Орлова Дарья, Павлов Алексей и Шумов Владимир, Владимирская обл., г. Струнино, школа № 11, учитель **Волкова Т.П.**;

— Прокопьев Максим, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Задача «Стоянка машин»

Напомним условие: «На стоянке стоят пять машин. Известно, что «Жигули» стоят перед «Волгой», «Ауди» — после «Тойоты», «Волга» — перед «Мерседесом», «Мерседес» — перед «Тойотой». В каком порядке стоят машины на стоянке?».

Ответ: «Жигули», «Волга», «Мерседес», «Тойота», «Ауди».

Правильный ответ прислали:

— Ашкин Матвей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Богославцев Игорь и Гриценко Данил, школа № 1 г. Зеленокумска Ставропольского края, учитель **Букина Е.Ю.**;

— Востриков Максим и Чуркин Владислав, г. Пенза, школа № 73, учитель **Диков А.В.**;

— Невоструев Семён, Орлова Дарья, Павлов Алексей и Шумов Владимир, Владимирская обл., г. Струнино, школа № 11, учитель **Волкова Т.П.**

Ответы за задания, опубликованные в журнале «Мир информатики», выпуск № 32 (март 2019 г.), представили ученики школы № 1 г. Зеленокумска Ставропольского края (учитель **Букина Е.Ю.**):

— Абкаров Михаил школа (задача «Как проехать из А в В?»);

— Азаров Данил, Груздева Дарья и Турченко Алика (задача «Четыре сообщения»);

— Бабкин Сергей, Гришко Ксения, Никитин Илья и Олейникова Ангелина (головоломка «Составить слово»);

— Богославцев Игорь (задачи «Три брата» и «Экзамены, экзамены...»);

— Литвиненко Елизавета и Хижняк Милена (кроссворд);

— Малыгин Сергей (задача «Дело об украденном компьютере»);

— Ткаченко Дмитрий (головоломка «Путешествие по буквам»).

Ученики этой же школы Аксенова Кристина, Емельченко Анастасия и Кашина Екатерина правильно решили ребусы по информатике, а Миронян Яна — числовые ребусы в пятеричной системе счисления.

Числовые ребусы в пятеричной системе счисления также решили:

— Абдувахидова Алина, Абдувахидова Софья и Куклева Лидия, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Андрукевич Максим, Дусаева Элина, Егоров Даниил, Кирсанов Данил и Кравцов Владислав, Республика Коми, г. Усинск, Центр дополнительного образования детей, «Школа программистов», куратор **Демяхина О.В.**;

— Боровиков Иван, г. Пенза, школа № 73, учитель **Диков А.В.**

Ответы за задания, опубликованные в одном из предыдущих выпусков журнала «Мир информатики», представили учащиеся «Школы программистов» Центра дополнительного образования детей г. Усинска, Республика Коми (куратор **Демяхина О.В.**):

— числовые ребусы в четверичной системе счисления: Ремезов Демьян;

— японские головоломки судоку: Андрукевич Максим и Ремезов Демьян;

— головоломки «Азбука информатики в загадках. Часть 3» и «Необычный ребус»: Андрукевич Максим.

Надписи на шкатулках

Д.Б. Невидимый,
Москва

Решите, пожалуйста, три логические задачи.

1. В одной из трех шкатулок (золотой, серебряной и бронзовой) находится портрет. На каждой шкатулке написано одно высказывание:

золотая	серебряная	бронзовая
Портрет в этой шкатулке	Портрет не в этой шкатулке	Портрет не в золотой шкатулке

Определите, где находится портрет, если известно, что из трех высказываний истинно только одно.

2. Теперь из трех высказываний на крышке шкатулок по крайней мере одно истинно по крайней мере и одно ложно. Определите, где находится портрет, если на шкатулках написано:

золотая	серебряная	бронзовая
Портрет не в серебряной шкатулке	Портрет не в этой шкатулке	Портрет в этой шкатулке

3. Теперь на каждой шкатулке написано по два высказывания, из которых ложно не более чем одно:

золотая	серебряная	бронзовая
Портрет не в этой шкатулке	Портрет не в золотой шкатулке	Портрет не в этой шкатулке
Портрет написан художником из Венеции	Портрет написан художником из Флоренции	Портрет в серебряной шкатулке

Определите, в какой шкатулке находится портрет и кто его написал.

Бусинки с буквами

Имеются бусинки с буквами. Нужно сформировать трехбуквенные цепочки, соблюдая следующий правила:

1) на первом месте в цепочке стоит одна из бусин А, Б, В;

2) на втором – одна из бусин Б, В, Г;

3) на третьем – одна из бусин А, В, Г, не стоящая в цепочке на первом или втором месте.

Сколько таких цепочек можно сформировать?

Три пирата

У пиратов А, Б и В состоялся такой разговор:

А: «У Б – 2 глаза».

Б: «У В – 2 глаза».

В: «У А – 2 глаза».

А: «У нас 2 глаза на троих».

Б: «У нас 3 глаза на троих».

В: «У нас 4 глаза на троих».

Оказалось, что каждый соврал столько раз, сколько у него глаз. Сколько глаз у каждого из пиратов?

Автор задачи — Егор Бакаев

Поддельная таблетка

Среди 11 таблеток есть одна поддельная, которая отличается от настоящих только массой, но в какую сторону и насколько — неизвестно. За какое минимальное число взвешиваний таблеток на чашечных весах без гирь можно определить, какая таблетка тяжелее — поддельная или настоящая? Ответ обоснуйте описанием соответствующего алгоритма решения задачи.

Числовой ребус «Два удара»

В приведенном ниже числовом ребусе одинаковыми буквами зашифрованы одинаковые цифры, разными буквами – разные цифры. Решите, пожалуйста, его.

```
  У Д А Р
+ У Д А Р
-------
Д Р А К А
```

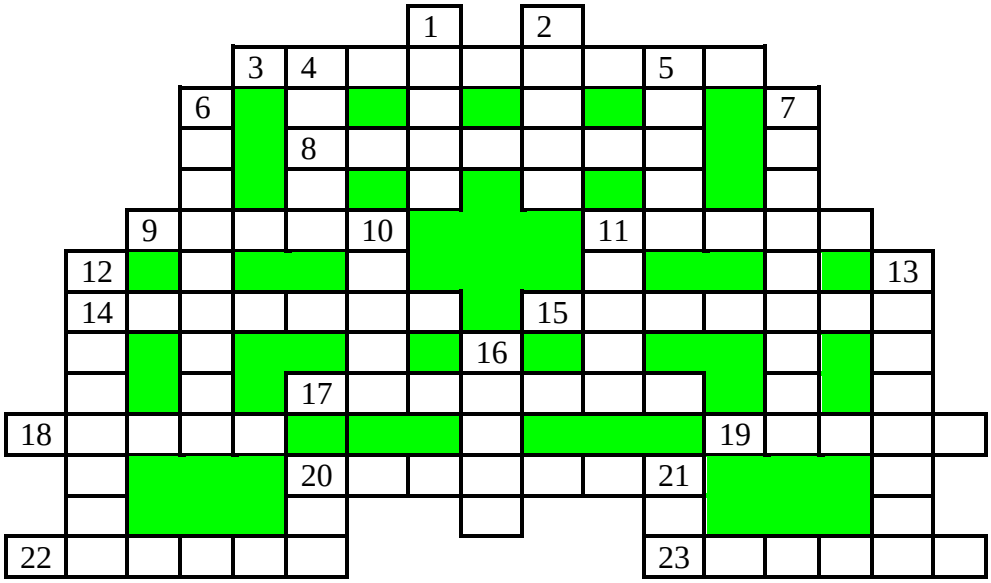
Три фигуры

Три фигуры — Треугольник, Круг и Квадрат живут в трех домиках: домик с высокой крышей, но маленьким окном; с высокой крышей и большим окном; с низкой крышей и большим окном. Треугольник и Круг живут в домиках с большими окнами, а Круг и Квадрат - в домиках с высокой крышей. Определите, кто в каком домике живет.

Задача предназначена для учеников 1–7-х классов.

Кроссворд

Решите, пожалуйста, кроссворд:



По горизонтали

3. Таблица соответствий между экранными изображениями символов и соответствующими им числами (например, КОИ-8).

8. Единица измерения информации, равная 1024 битам.

9. Изобретатель системы кодирования информации, использующей два символа — точку и тире.

11. Реакция объекта на воздействие или запрос.

14. Часть стандартного устройства для ввода информации в компьютер.

15. Вид связи.

17. Устройство, машина или система, выполняющие действия по заданной программе без участия человека.

18. В базах данных — таблица с результатами расчетов, а также документ, содержащий данные о годовых результатах деятельности компании (но ё = е).

19. Марка персональных компьютеров и калькуляторов, выпускавшихся в СССР, а также мельчайшая частичка горящего или раскаленного вещества.

20. Участок магнитного диска в виде двух концентрических окружностей, образуемый при разметке диска.

22. Один двух режимов ввода текста в текстовых редакторах.

23. Устройство управления работой шины персонального компьютера (так также называют футбольного судью).

По вертикали

1. Величина, с помощью которой осуществляется счет.

2.

4. Неисправность в работе компьютера.

5. Сторона прямоугольного треугольника.

6. Число, определяющее систему счисления.

7. Совокупность средств взаимодействия программы и пользователя.
10. Советский академик, один из основоположников школьной информатики.
11. Буква ω.
12. Наука, изучающая звук.
13. Глобальная компьютерная сеть.
16. Управляемая по программе и имитирующая действия человека машина.
20. Цифра десятичной системы счисления.
21. Язык программирования, названный в честь первой женщины-программиста.

Числовые ребусы в пятеричной системе счисления. Часть 2

В приведенных ниже ребусах буквами зашифрованы цифры чисел, записанных в пятеричной системе счисления (одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры; звездочка может быть любой цифрой). В разных ребусах одной и той же буквой могут быть зашифрованы разные цифры. Решите эти ребусы.

1.	_ **A B C** **D** **D D**	**2.**	+ **D D** **C** * * 2	**3.**	+ **P P** **R** **R** 0

Илья Муромец и Змей Горыныч

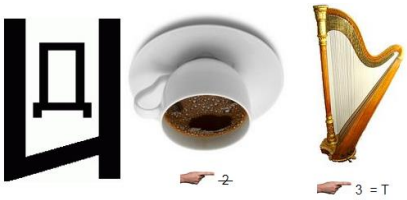
Перед вами — фрагмент сказки: «Срубил Илья Муромец Змею Горынычу голову, а взамен две выросли. Срубил 2 — выросли 4, срубил 4 — выросли 8, срубил 8 — выросли 16, срубил 16 — выросли 32, срубил 32 — выросли 63, срубил 63 — выросли 128, срубил 128 — выросли 256. А как срубил Илья 256 голов, тут и настал Змею Горынычу конец, потому что был Горыныч восьмиразрядный!»

Какие *три* ошибки имеются в приведенном фрагменте?

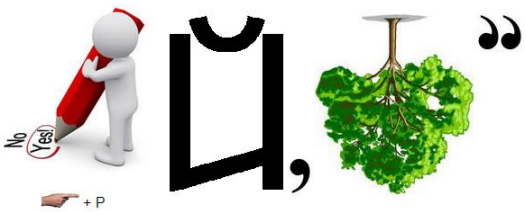
Три непростых ребуса

Решите, пожалуйста, ребусы, связанные с информатикой.

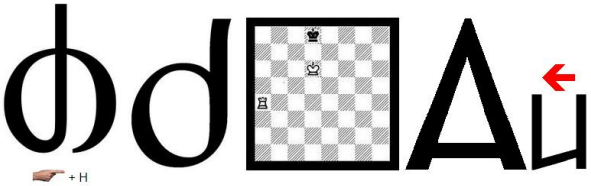
Ребус № 1



Ребус № 2



Ребус № 3



Ответы присылайте в редакцию (можно решать не все ребусы).

Итоги конкурса № 20

Напомним, что в качестве заданий этого конкурса было предложено ответить ещё на ряд вопросов, связанных с так называемыми «логическими операциями над числами» (статья с соответствующим названием было опубликована в августовском 2018 года выпуске журнала «Мир информатики»).

Во всех случаях дано целое двоичное число x, которое в памяти компьютера представляется как последовательность из восьми битов (может быть, с начальными нулями).

1. Какую логическую операцию и с какой маской надо выполнить над этим числом, чтобы обнулить крайний справа единичный бит этого числа (т. е. в двоичном представлении числа превратить крайнюю справа единицу в ноль). Например, при x = 01011000 необходимо получить 01010000.

Ответ

Применить заданному числу логическую операцию конъюнкции с маской, равной этому же числу, уменьшенному на 1. Например, если заданное число равно 01011000, то после вычитания из него 1 получится 01010111, а в результате конъюнкции:

AND	0	1	0	1	**1**	0	0	0
	0	1	0	1	0	1	1	1
	0	1	0	1	**0**	0	0	0

Логика действий здесь такая. Цифры исходного число до крайнего справа единичного бита не должны измениться. Это можно сделать, если применить к этим разрядам логическую операцию **AND** с маской, равной числу из тех же цифр. А из остальных цифр, которые имеют вид 1000, надо получить 0000. Это можно сделать, если маска будет соответствующих разрядов равна 0111. Получить же такую маску можно, вычтя из заданного числа 1.

2. Как, используя логические операции, проверить, является ли оно степенью числа 2?

Ответ

Так как речь в задаче идет о проверке значения, то остается применить к заданному числу какую-то логическую операцию и проверить результат на равенство нулю[11]. Заданное число x в двоичном виде выглядит так:

10…0.
k нулей

Получить ноль можно, применив к числу x логическую операцию **AND** с маской из k единиц. Такая маска может быть получена следующим образом: $x - 1$.

3. Как, используя логические операции, проверить, имеет ли оно вид $2^k - 1$?

Ответ

Решение этой задачи как бы «противоположно» решению предыдущей. Получить ноль (для проверки) можно, применив в заданному числу, которое должно иметь вид:

11…1,
k единиц

логическую операцию **AND** с маской вида:

10…0
k нулей

[11] Проверять на равенство единице и двойке нельзя, так эти числа сами являются степенью числа 2.

Такая маска может быть получена следующим образом: $x + 1$, где x — заданное число.

4. Какую логическую операцию и с какой маской надо выполнить над этим числом, чтобы выделить крайний справа единичный бит этого числа (т. е. получить 8 бит, среди которых единственная единица находится в том разряде, в котором в заданном числе находился крайний справа единичный бит). Например, при x = 01011000 необходимо получить 00001000.

Ответ

Здесь рассуждения могут быть такими. Получить число из одних нулей можно, применив к заданному числу логическую операцию **AND** с маской, равной числу, цифры которого противоположны соответствующим цифрам заданного числа (его, в свою очередь, можно получить с помощью операции **NOT**). А чтобы в результате была единственная единица в том разряде, в котором в заданном числе находился крайний справа единичный бит, нужно, чтобы при упомянутой чуть выше операции **AND** в этом разряде в маске также была единица. Такую маску можно получить, перед использованием операции **NOT** вычтя из заданного числа 1. Итак, этапы решения задачи:

1) вычесть из заданного числа 1. Например, если заданное число равно 01011000, то после вычитания из него 1 получится 01010111;

2) применить к результату отрицанию **NOT** — получится 10101000;

3) полученное в пункте 2 число использовать в качестве маски в операции **AND** с заданным числом:

	0	1	0	1	**1**	0	0	0
AND	1	0	1	0	1	0	0	0
	0	1	0	1	**1**	0	0	0

Второй способ иллюстрируется следующей таблицей:

x	0	1	0	1	1	**0**	**0**	**0**
NOT x	1	0	1	0	0	1	1	1
NOT x + 1	0	1	0	1	1	0	0	0
x **AND** (**NOT** x + 1)	0	1	0	1	**1**	0	0	0

5. Какую логическую операцию и с какой маской надо выполнить над этим числом, чтобы выделить завершающие нулевые биты этого числа в виде единиц? (Например, при x = 01011000 необходимо получить 00000111.)

Ответ

Задача может быть решения несколькими способами.

1. Логическая формула для получения требуемого результата имеет вид: **NOT** x **AND** (x – 1).

Например, при x = 01011**000**, имеем **NOT** x = 10100111, x – 1 = 01010**111**, **NOT** x **AND** (x – 1) = 00000111, при x = 11011**00**: имеем **NOT** x = 00100011, x – 1 = 11011011, **NOT** x **AND** (x – 1) = 00000**11**.

2. Вторая логическая формула, по которой можно получить требуемый результат, выглядит так:

NOT (x **OR NOT** (x – 1)).

Пример ее использования:

x	0	1	0	1	1	**0**	**0**	**0**
x – 1	0	1	0	1	0	1	1	1
NOT (x – 1)	1	0	1	0	1	0	0	0
x **OR NOT** (x – 1)	1	1	1	1	1	0	0	0
NOT (x **OR NOT** (x – 1))	0	0	0	0	0	**1**	**1**	**1**

3. Имеется также вариант, во многом аналогичный описанному только что. Для него формула имеет вид:

NOT (*x* **OR** (**NOT** *x* + 1)).

x	0	1	0	1	1	**0**	**0**	**0**
NOT *x*	1	0	1	0	0	1	1	1
NOT *x* + 1	1	0	1	0	1	0	0	0
x **OR** (**NOT** *x* + 1)	1	1	1	1	1	0	0	0
NOT (*x* **OR** (**NOT** *x* + 1))	0	0	0	0	0	**1**	**1**	**1**

4. Четвертая формула: (*x* **AND NOT** (*x* – 1)) – 1.

x	0	1	0	1	1	**0**	**0**	**0**
x – 1	0	1	0	1	0	1	1	1
NOT (*x* – 1)	1	0	1	0	1	0	0	0
x **AND NOT** (*x* – 1)	0	0	0	0	1	0	0	0
(*x* **AND NOT** (*x* – 1)) – 1	0	0	0	0	0	**1**	**1**	**1**

Здесь также возможен вариант:

x	0	1	0	1	1	**0**	**0**	**0**
NOT *x*	1	0	1	0	0	1	1	1
NOT *x* + 1	1	0	1	0	1	0	0	0
x **AND** (**NOT** *x* + 1)	0	0	0	0	1	0	0	0
x **AND** (**NOT** *x* + 1) – 1	0	0	0	0	0	**1**	**1**	**1**

6. Какую логическую операцию и с какой маской надо выполнить над этим числом, чтобы выделить крайний справа единичный бит и все завершающие нулевые биты этого числа в виде последовательности единиц? (Например, при *x* = 01011000 необходимо получить 00001111.)

Ответ

Для решения задачи необходимо к заданному числу применить логическую операцию **XOR** с маской, равной этому же числу, уменьшенному на 1:

x	0	1	0	1	**1**	**0**	**0**	**0**
x – 1	0	1	0	1	0	1	1	1
x **XOR** (*x* – 1)	0	0	0	0	**1**	**1**	**1**	**1**

7. Какую логическую операцию и с какой маской надо выполнить над этим числом, чтобы распространить крайний справа единичный бит? (Например, при *x* = 01011000 необходимо получить 01011111.)

Ответ

Отличие решения этой задачи от предыдущей заключается в использовании операции «обычной» дизъюнкции вместо исключающей дизъюнкции:

x	0	1	0	1	**1**	**0**	**0**	**0**
x – 1	0	1	0	1	0	1	1	1
x **OR** (*x* – 1)	0	1	0	0	**1**	**1**	**1**	**1**

Победителями конкурса признаны:

— Веремеев Владимир, г. Пенза, школа № 73, учитель **Диков А.В.**;

— Крайнов Сергей, Владимирская обл., г. Александров, школа № 13, учитель **Семенкович А.О.**;

— Куклева Лидия, Владимирская обл., г. Струнино, школа № 11, учитель Волков Ю.П.;

— Старостина Наталья, Владимирская обл., г. Александров, школа № 14, учитель **Солодова Е.Н.**;

— Шумилин Максим, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Все перечисленные читатели будут награждены дипломами. Поздравляем!

Конкурс № 22

В качестве задания этого конкурса предлагаем решить задачу, которая была опубликована в одном из предыдущих номеров нашего журнала.

Задача «Рыбалка»

Четверо ребят удили рыбу. Всего они поймали шесть рыб. Один поймал три рыбы, второй — две рыбы, третий — одну рыбу, четвертый — ни одной. Известно, что:

1) тот, кто поймал две рыбы, в качестве насадки пользовался не червями и не живцом;

2) тот, кто ловил на живца, поймал меньше рыб, чем Федор;

3) лучшей насадкой в тот день оказались мухи — на них было поймано больше всего рыбы;

4) Толик пользовался в качестве насадки опарышами;

5) Сергей поймал больше, чем тот, кто ловил на живца, но не больше всех;

6) четвертого рыболова зовут Иван.

Кто из них сколько рыб поймал и что каждый использовал в качестве насадки?

Ответ (с обоснованием!) отправьте в редакцию по адресу: *mirinfo@infojournal.ru*

Пожалуйста, укажите в письме свои фамилию и имя, населенный пункт, номер школы и фамилию, имя и отчество учителя информатики.

Срок представления ответов — 30 сентября.

Японский уголок

Два судоку

Решите, пожалуйста, две японские головоломки “судоку”:

1) простую:

2	4				5	3		8
			2			6		5
	6		8		7	9		
	1	4		8			6	2
	5	2	3		4	8	9	
8	9			5		4	3	
		9	5		1		8	
1		7			6			
6		5	9				4	3

2) сложную:

			9					6
1			7	2	5	3		8
9					6	5		7
2	9		6	7				
			2					
		8		5		4	1	
		4			1	6	8	
	6					9		
					7		5	

Ответы (можно не на все головоломки) присылайте в редакцию.