ОБЗОР РЕШЕНИЙ ЗАДАЧ ЗАНЯТИЯ 1.

Приведенные ниже решения, безусловно, хороши. Но если мы получим от вас лучшие решения, то мы их, конечно же, опубликуем.

Решения задач 1, 2, 8, 10 и 13 мы напечатаем в следующих номерах. Нам вначале хочется ознакомиться с вашими версиями. Так что "пишите письма".

Задача 3.

Ваш знакомый живет в стандартном двенадцатиэтажном доме в квартире 87. На каком этаже может располагаться его квартира? (На лестничной площадке одно и то же число квартир). Вообще, как быстро определить, может ли квартира с данным номером находиться на данном этаже?

Уровень 1. После некоторого размышления можно увидеть, что на нуль никакое другое число не делится, кроме самого нуля, который, в свою очередь, делится на любое число. Предположим, что на каждом этаже помещается n квартир. Остатки от деления на *n* номеров квартир, расположенных на одном этаже, будут соответственно 1, 2 .. *n*-2, *n*-1, 0.

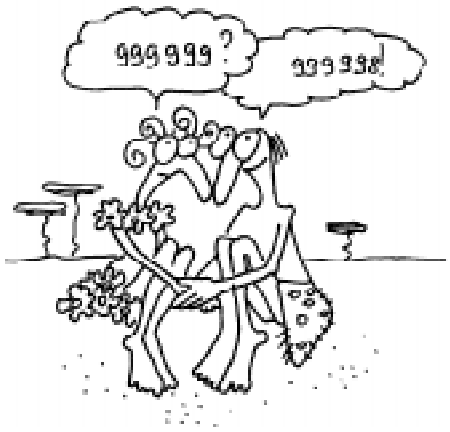
При переходе на следующий этаж все номера квартир увеличиваются на *n*, и при этом остаток от деления на *n* не меняется. Зато меняется частное от деления на *n* с остатком. Для квартир на первом этаже оно будет равняться нулю для всех квартир, кроме квартиры с номером *n*. Почему такая несправедливость? Вот если

бы нумерация квартир и этажей начиналась с нуля, то у нас не было бы никаких проблем – частное от деления номера квартиры на n в точности совпадало бы с номером этажа, на котором она находится. Если хочешь быть счастливым, будь им – давайте вычтем и прибавим там, где нужно, эту единичку.

Тогда для квартиры с номером p номер этажа q будет равняться

$q=(p-1)\mathrm{div}n+1$

Следующий вопрос – может ли квартира с номером p находиться на этаже с номером q? Предположим, что может. Тогда $p-1=n(q-1)+r$, где r – это номер квартиры на своем этаже (от нуля до $n-1$) и, следовательно, он должен быть меньше n. Таким образом, нам нужно подобрать такое n, чтобы r получилось как можно меньше. Но несложно заметить, что самое маленькое r мы получим, если разделим $p-1$ с остатком на $q-1$. Если при этом r окажется меньше n, то, значит, мы сумели расположить квартиру на требуемом этаже. В противном случае квартира не может располагаться на q-том этаже.



Задача 4.

Язык племени мумбу-юмбу состоит из шестибуквенных слов, составленных из букв {А, Б, В, Г, Д, Е, Ж, З, И, К}. В Оксфорде издан полный словарь слов этого языка (упорядоченных по алфавиту:
АААААА, ААААБ, ААААВ,..., КККККИ, КККККК.

На каждой странице словаря помещается 15 слов.

На каких страницах и в каких строках находятся слова ДЕКАДА и ЗАБАВА?

Уровень 1. Так уж получилось, что букв в племени мумбу-юмбу ровно столько, сколько и цифр в десятичной системе счисления. Это дает нам возможность каждую букву заменить на соответствующую цифру – А на 0, Б на 1, ...К на 9. Таким образом все слова превратятся в числа от нуля (000000) до 999999. Достаточно выяснить частное от деления такого слова на 15 с остатком, чтобы узнать сколько полных страниц набралось из слов, стоящих раньше данного, если смотреть в алфавитном порядке. А когда мы увеличим это число на единицу, мы узнаем номер страницы, на которой записано слово:

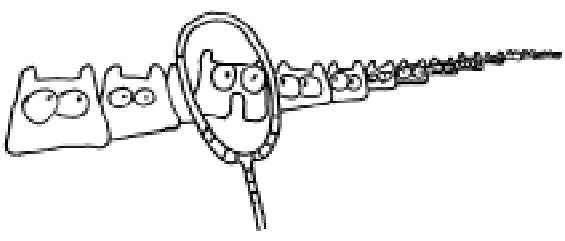
ДЕКАДА mod 15 + 1 = 459040 mod 15 +1 = 30603

ЗАБАВА mod 15 + 1 = 701020 mod 15 +1 = 46735

Задача 5.

Найдите цифру с номером *n* в последовательности 01234567891011121314151617181920... записанных подряд натуральных чисел.

Уровень 1. Для начала признаемся в неточной формулировке задачи в предыдущем номере. Дело в том, что последовательность 012345678910111213141516171819... не является последовательностью записанных подряд натуральных чисел. Как известно, натуральные числа начинаются с единицы. Поэтому следовало написать: последовательность записанных подряд целых чисел, начиная с нуля.



Теперь приступим к разбору. Сначала разберемся, со “сколько значными” числами нам придется работать. Если номер n меньше десяти, значит соответствующее число состоит из одной цифры.

В противном случае оно состоит из двух или более цифр. В таком случае да-

вайте выкинем из рассмотрения однозначные числа, то есть уменьшим число n на 10 и будем считать, что последовательность начинается с десяти. Наше число 1999 таким образом превращается в 1989.

Давайте таким образом уменьшим наш ряд по максимуму. В итоге должна остаться последовательность записанных подряд чисел одинаковой длины, которые начинаются со степени десятки.

Каждый шаг уменьшения ряда сводится к тому, чтобы определить, можем ли мы выкинуть ведущий кусок ряда из чисел одинаковой длины k так, чтобы в получившейся последовательности осталась интересующая нас цифра с номером n.

После первого шага (удаление всех однозначных чисел) k=2.

Сколько существует k-значных чисел? Их количество равняется 10^k-10^{k-1}. Следовательно, количество цифр в них p будет равняться $k(10^k\text{-}10^{k-1})$. Если наш номер n больше или равен p, то нужно уменьшить его на p и считать, что наш ряд начинается с 10^k (увеличиваем k на единицу). Если $n<p$, то это значит, что мы уже уменьшили наш ряд насколько возможно.

Посмотрим, что происходит с числом 1989.

n	k	Вырезаемый кусок	Его длина
1989	2	двузначные	90*2
1809	3	трехзначные	900*3>n

Итак, нам нужно найти цифру с номером n (1809) в последовательности подряд выписанных k-значных (трехзначных) чисел.

Теперь мы можем определить число A, в котором находится искомая цифра. Его номер в последовательности равен $n$div$k$ (если считать с нуля). Следовательно, A=$n$div$k$+10^{k-1} (так как 10^{k-1} – первое число в последовательности. То есть само число A=1809div3+100=703.

Далее, нам нужно определить цифру с номером $n$mod$k$ в k-значном числе A. В нашем случае это нулевая цифра числа 703, – то есть семерка.

Уровень 2.

```
var n, dec : longint;  {в переменной dec хранится
k-тая степень десятки}
    A, k, p, i: integer;   {p - количество
k-значных чисел}
begin
  readln(n);
  k:=1;
  dec:=10;
  p:=10;

  while p >= n do
    begin
      n:=n-p;                {уменьшаем ряд}
      inc(k);              {k - число цифр в числах в
      dec:=dec*10;        начале ряда}
      p:=dec - dec div 10;
    end;

  A := n div k + dec div 10;

  dec := 1;
  for i:= 1 to k - n mod k do dec:=dec*10;
{вычисляем 10^(k - n mod k)}
  writeln((A mod dec) div (dec*10));
end.
```

Задача 6.

Для любого натурального числа алгоритм совершает следующие операции: отделяет от числа первую цифру и прибавляет ее к числу из оставшихся цифр. Процесс продолжается до тех пор, пока в числе останется одна цифра. Определить результат работы алгоритма для чисел вида $\underbrace{88\ldots8}_{n}$ при различных значениях *n*.

Уровень 1. Обычно в таких задачах следует искать величину, которая от шага к шагу не меняется. Она называется инвариантом. В данном случае инвариантом является остаток от деления на девять.

Посмотрим, что происходит с числом при каждом шаге алгоритма в арифметике остатков. Во-первых мы “отрезаем” от него первую цифру – то есть вычитаем из него первую цифру помноженную на какую-то степень десятки. В арифметике остатков по модулю 9 любая степень десятки равна единице. Поэтому, "отрезание" первой цифры числа в арифметике остатков равносильно вычитанию этой цифры. А затем (во-вторых) мы прибавляем к нему эту же цифру. В итоге остаток от деления на 9 не меняется.

На последнем шаге алгоритма мы можем получить число от 1 до 9 (потому что остается только одна цифра). У всех этих чисел различные остатки от деления на 9. Поэтому, зная остаток исходного числа, мы можем однозначно указать результат.

А остаток исходного числа легко вычислить по сумме цифр. Он равен $8n \bmod 9$.

Уровень 2.

Напишем программу, которая выдает все промежуточные результаты для чисел из не более чем 50 цифр. Здесь нет ничего сложного. Нужно просто аккуратно выполнить все действия. Цифру будем прибавлять столбиком – как учили в первом классе.

```
var a: array[0..49] of 0..9;
    n, i:1..50;
    p: byte;  {число "в уме"}
begin
  readln(n);
  {будем записывать цифры в массив так, чтобы номер ячейки
  совпадал со степенью десятки в данном разряде}
  for i:=n-1 downto 0 do readln(a[]);
  while n>1 do
    begin
      p:=a[n-1];  {вычеркиваемая цифра}
     dec(n);
      j:=0;       {позиция в числе}
      while p>0 do
        begin
          p:=p+a[j];
          a[j]:=p mod 10;
          p:=p div 10;
          inc[j];
           if (j>n-1) and (p>0) {а вдруг мы вылезли}
             tnen begin         {за пределы числа}
                    a[j]:=0;
                    n:=j+1;
                  end;
        end;
       for i:=n-1 downto 0 do write(a[j]);
       writeln;
    end;
end.
```

Задача 7.

Известно, что любую дробь $\frac{a}{b}$, где *a* и *b* - натуральные числа, можно представить в виде цепной дроби. Опишите алгоритм. Представьте в виде цепной дроби $\frac{17}{239}$.

Уровень 1. Для тех, кто разобрался в теории, описанной в прошлом номере, алгоритм прост.

Шаг алгоритма: вычисляем целую часть от деления a на b и запоминаем в качестве очередного c. Вместо a записываем остаток от деления a на b. Если он равен 1, то запоминаем b в качестве последнего частного. Если нет, то меняем местами a и b и переходим к следующему шагу.

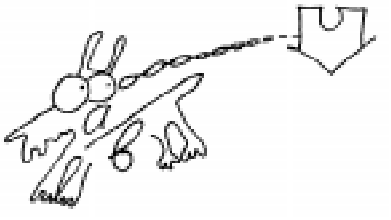
Вот что произойдет с числом $\frac{17}{239}$:

a	b	c	r
17	239	**0**	17
239	**17**	**14**	1

$$\frac{17}{239} = 0 + \frac{1}{14 + \frac{1}{17}}$$

Уровень 2.

Напишите программу, которая по данным числам a и b представляет дробь $\frac{a}{b}$ в виде цепной (a, b<1000000).



```
var a, b, r, n: integer;
    c: array[0..200];
begin
  readln(a);
  readln(b);
  n:=0;  {количество элементов в c}
  r:=5;  {остаток - любой, лишь бы не единица}
  while r<>1 do
    begin
      c[n]:=a div b;
      inc(n);
     r:=a mod b;
      b:=a;
      a:=r;
    end;
  c[n]:=a;
  inc(n);

  writeln(n);
  for i:=0 to n-1 do writeln(c[i]);
end.
```

Задача 9.

Уровень 1. **Обобщить алгоритм Евклида с делением на 2 и вычитанием для наборов из более чем двух чисел. Применить обобщенный алгоритм к набору {72; 84; 132; 144}.**

ШАГ 1. Положим НОД равным единице. Пока все числа из набора четные, будем их всех делить на 2 и домножать на эту двойку НОД. Когда хотя бы одно из чисел станет нечетным, переходим к шагу 2.

ШАГ 2. Делим все четные числа набора на 2 до тех пор, пока деление возможно.

ШАГ 3. Если в наборе более одного ненулевого числа, то заменяем большее из чисел набора разностью со следующим по величине и переходим к шагу 2, иначе умножаем НОД на полученный результат и заканчивам работу.

				НОД
72	**84**	**132**	**144**	1
36	**42**	**66**	**72**	2
18	21	33	**36**	4
9	**21**	**33**	9	4
9	21	**12**	9	4
9	**21**	3	9	4
9	**12**	3	9	4
9	3	3	**9**	4
9	**3**	3	0	4
6	3	3	0	4
3	**3**	3	0	4
3	0	**3**	0	4
3	0	0	0	12

Уровень 2. **Доказать, что если одно из чисел a, b четное, а другое нечетное, то следующее преобразование сохра-**

няет множество общих делителей *a* и *b*:
ЕСЛИ ровно одно из чисел *a, b* четное
ТО поделить его на 2
ИНАЧЕ заменить большее из чисел *a, b* их разностью

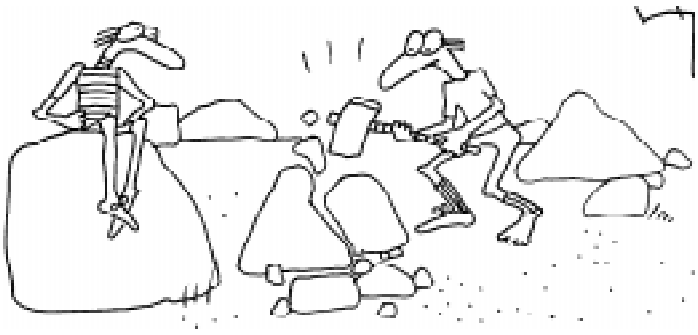
Понятно, что при делении одного из чисел на 2 новых общих делителей не появится.

Что же будет со старыми? Предположим, d является одним из общих делителей. Пусть число a четно. Тогда $a=2dk$ (d не может делиться на два, так как одно из чисел не кратно двум, а d – общий делитель). Видно, что от деления числа a на 2 оно не перестанет делиться на d.

Следовательно, при первом преобразовании множество общих делителей не изменится. То, что замена числа разностью не меняет множества общих делителей доказывалось на первом занятии.

Задача 11.

Из кучки камней двое играющих по очереди берут 1, 2 или 3 камня. Проигрывает тот, кто берет последний камень. Предположим, что всего камней *N*. Кто из игроков имеет выигрышную стратегию? Опишите ее. А если выигрывает взявший последний камень?



Уровень 1. Давайте определимся, – что такое выигрышная стратегия. Говорят, что игрок имеет выигрышную стратегию, если на любые ходы противника он может отвечать так, что придет к победе.

Выигрышная позиция – это позиция, в которой у игрока, который ходит первым, существует выигрышная стратегия. Проигрышная позиция – это позиция, в которой любые ходы приводят к проигрышу (при умном противнике).

Разберем позиции в нашей игре. Понятно, что ситуация, когда остался 1 камень, является проигрышной. Игрок неминуемо терпит поражение.

Если позиция такова, что существует ход, приводящий к проигрышной позиции, значит она выигрышная. Действительно, делая такой ход, мы предоставляем противнику проигрышную позицию. Таким образом, ситуации, когда осталось 2, 3 или 4 камня, являются выигрышными.

Если любой ход приводит в выигрышную ситуацию, значит позиция проигрышная – мы любым ходом неминуемо приводим противника к хорошему расположению. Так, позиция с пятью камнями является проигрышной.

Таким образом, мы можем сосчитать любую игровую ситуацию (плюсами отмечены выигрышные):

N	1	2	3	4	5	6	7	8	9	10	11	...
	-	+	+	+	-	+	+	+	-	+	+	...

Вот табличка для игры, когда выигрывает взявший последний камень:

N	1	2	3	4	5	6	7	8	9	10	11	...
	+	+	+	-	+	+	+	-	+	+	+	...

Если в начале игры позиция выигрышная, выигрывает первый игрок, в противном случае – второй.

К задаче можно подойти и с другого конца – для того чтобы выиграть, нужно дополнять ход противника до четырех. Если он взял один камень, то мы три; если он -

два, то и мы два. Тогда остаток от деления на четыре после нашего хода будет один и тот же. В первом случае это единица, а во втором – ноль.

Задача 12.

Уровень 1. **Алиса, попав в Страну Чудес, забыла таблицу умножения. Она говорит: семью семь будет ... пятнадцать, девятью девять будет ... тринадцать.**

Определите в какой модульной арифметике такие правила умножения справедливы. Сколько всего таких арифметик?

Но семью семь на самом деле 49, а девятью девять – 81. Значит

$49 \equiv 15 \pmod{m}$

$81 \equiv 13 \pmod{m}$

Это означает, что 49 и 15, также как и 81 и 31, дают одинаковые остатки при делении на m. Другими словами, разность каждой пары делится на m, что можно записать так:

$34 \equiv 0 \pmod{m}$

$68 \equiv 0 \pmod{m}$

Следовательно, число m является одним из общих делителей чисел 34 и 68. Кроме того, это число должно быть больше 15, чтобы в модульной арифметике существовали числа 7, 9, 13 и 15. Этим условиям удовлетворяют числа 34 и 17.

Уровень 2. **Напишите программу, которая по введенным примерам таблицы умножения определяет, существует ли модульная арифметика, в которых эти примеры имеют место.**

```
var c: array[0..9];
    n, k, i: integer;
    maxn: integer;
    m: array[0..100] of integer;

function InputEquation: integer;
{Функция вводит уравнение и возвращает число,
которое, исходя из уравнения, должно равнять-
ся нулю по данному модулю. Также она записыва-
ет в переменную maxn значение наибольшего вве-
денного числа - нужно будет обеспечить его
существование}
  var s:string;
      a, b, c, mult, eq, err: integer;
  begin
    readln(s);

    mult:=2;
    while s[mult] <> '*' do inc(mult);
    eq:=mult+2;
    while s[eq] <> '=' do inc(eq);

    val(copy(s, 1, mult - 1), a, err);
    val(copy(s, mult + 1, eq - mult - 1), b, err);
    val(copy(s, eq+1, length(s) - eq), c, err);

    if a>maxn then maxn:=a;
    if b>maxn then maxn:=b;
    if c>maxn then maxn:=c;

    InputEquation:= abs(a * b - c);
  end;
```

```
begin
  maxn:=1;
  readln(n);
  k:=0;  {k - число ненулевых чисел в массиве c}
  for i:=0 to n-1 do  {ввод и обработка уравнений}
    begin
      c[i]:=InputEquation;
      if c[i]<>0 then inc(k); {записываем только}
    end;                    {ненулевые значения}

  while k>1 do   {вычисляем НОД массива c}
    begin
      if c[k-1]>c[k-2]
        then c[k-1]:=c[k-1] mod c[k-2]
        else c[k-2]:=c[k-2] mod c[k-1];
      if c[k-2]=0 then begin
                         c[k-2]:=c[k-1];
                c[k-1]:=0;
                           end;
      if c[k-1] = 0 then dec(k);
    end; {в итоге НОД находится в C[0]}

  k:=0;                {теперь k - число общих делителей}
  for i:=maxn+1 to c[0] do {или делителей НОД, больших maxn}
    if c[0] mod i = 0 {bo}
      then begin
             m[k]:=i;
             inc(k);
           end;

  writeln(k);
  for i:=0 to k-1 do writeln(m[i]);
end.
```

Задача 14.

Уровень 2. **Напишите программу, которая по заданным не более чем 25-значным числам *a, b, c* и *d* проверяет, верно ли, что *ab=cd*.**

```
type
  LongNumber = record
                 num: array[0..24] of 0..9;
                 len:integer;
               end;

var p: array[2..100]; {массив простых чисел}
    pcount: integer; {их количество}
    a, b, c, d: LongNumber;
    i, j: inyeger;
{процедура вводит длинное число}
procedure InputLongNumber(var n: LongNumber);
  var s: string;
  begin
    readln(s);
    for i:=0 to length(s)-1 do
      n.num[i]:=s[length(s)-i];
    n.len:=length[s];
  end;

{функция находит остаток от деления длинного числа на k}
function Residual(var n: LongNumber;k: integer): integer;
  var d, r, i :integer;
  begin
    r:=0;
    d:=1;
```

```
    for i:=0 to n.len-1 do
      begin
        r:=(r+d*n.num[i]) mod k;
        d:=d*10 mod k;
      end;
   Residual:=r;
  end;

begin
  for i:=2 to 100 do p[i]:=i;
  for i:=2 to 10 do   {метод нахождения простых чисел}
    begin              {называется Эратосфеновым решетом}
      if p[i] = 0 then continue;
      for j:=2 to 100 div i do p[i*j]:=0;
    end;
  pcount:=0;
  for i:=2 to 100 do {выкидываем из массива нули}
    if p[i]<>0 then begin
                      inc(pcount);
                      p[pcount+1]:=p[i];
                    end;

  InputLongNumber(a);
  InputLongNumber(b);
  InputLongNumber(c);
  InputLongNumber(d);

 eq: = true;
 for i:=2 to pcount+1 do {проверка равенства}
   if (Residual(a,p[i]) * Residual(a,p[i])-
       Residual(a,p[i]) * Residual(a,p[i])) mod p[i]<>0
     then begin
            eq:=false;
            break;
          end;

  if eq then writeln('Да')
        else writeln('Нет');
end.
```

Романовский Иосиф Владимирович, доктор физ.-мат. наук, профессор СПбГУ.

Черкасова Полина Геннадьевна, методист заочной школы современного программирования.

НАШИ АВТОРЫ