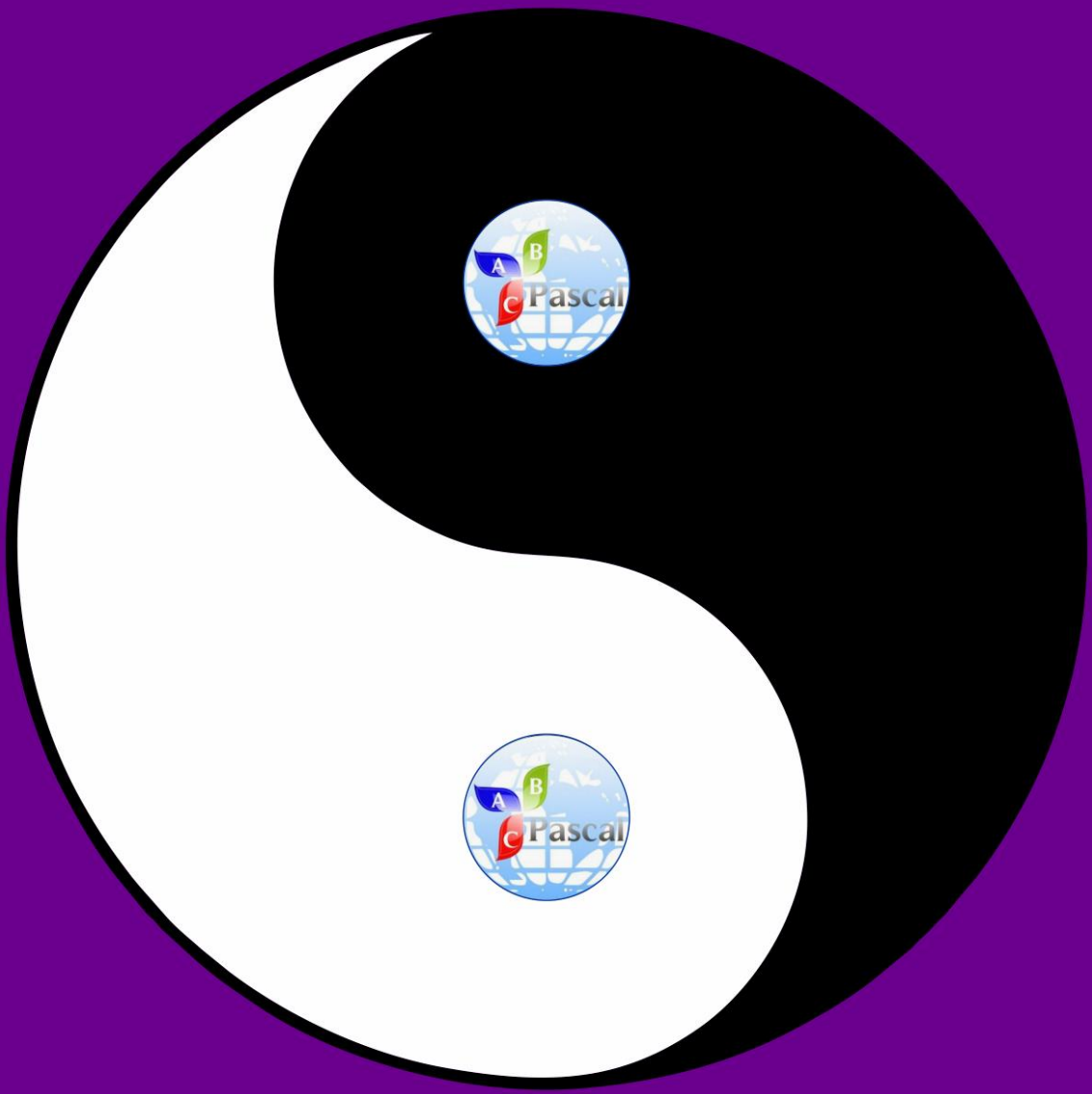
Рубанцев Валерий

Развивающее программирование

Увлекательная математика с паскалем

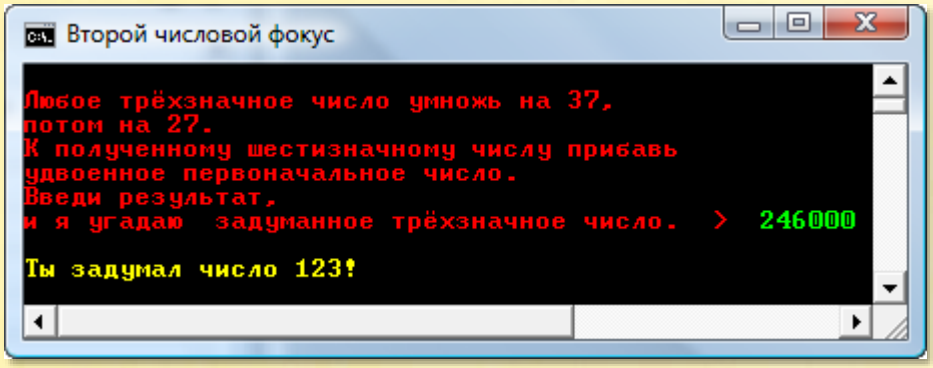


PascalABC.NET

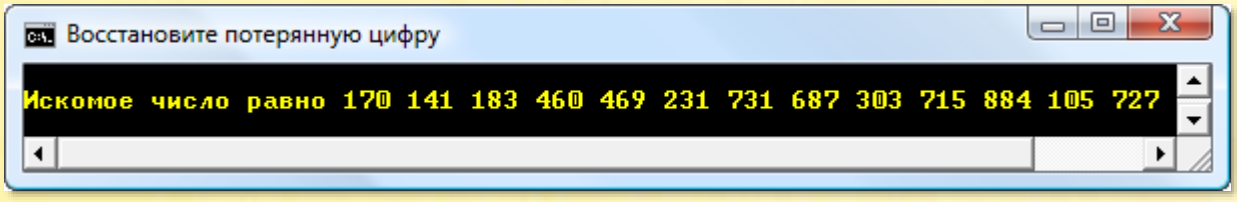
От автора

В книге *Решение задач на языке паскаль* мы решали исторические математические задачи всех времён и народов. А в этой книге мы будем решать исключительно отечественные головоломки. Я остановил свой выбор на авторах занимательных задач, которые хорошо известны каждому любителю математики. Это Фёдор Фёдорович Нагибин (*Математическая шкатулка*), Борис Анастасьевич Кордемский (*Удивительный мир чисел*) и Анатолий Павлович Савин (*Занимательные математические задачи*).

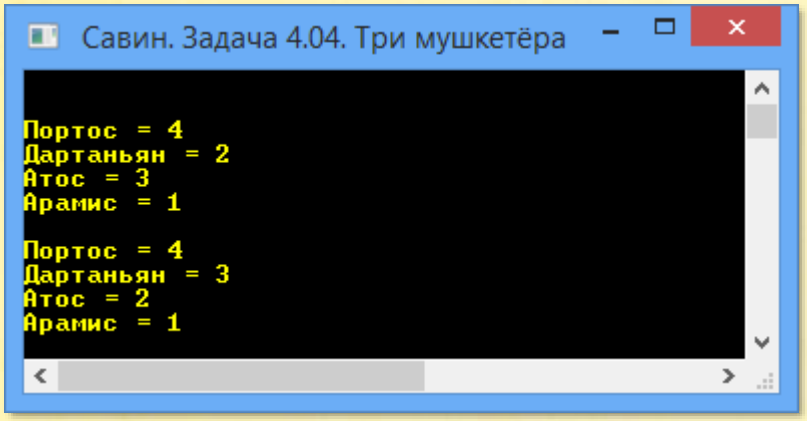
Вы научитесь показывать математические фокусы, решать числовые ребусы, пересчитывать кубики, палочки и страницы, освобождать лягушек, делить путёвки и ящики, с выгодой ходить на базар, кормить кашалота, продавать книги, решать рисованные задачи-комиксы...



Математический фокус



В поисках цифры



Ответ на математический комикс

Решая задачи, вы укрепите умения и навыки в применении таких элементов языка *PascalABC.NET,* как:

- Числовые типы данных – *integer, int64, double*
- Логический тип *boolean*
- Арифметические операции
- Простейшие математические функции
- Переменные (глобальные и внутриблочные) и константы
- Операторы объявления и присваивания
- Комбинированные операторы присваивания
- Логические операторы и выражения
- Условные логические операторы *if, if – else*
- Циклы *for, while, repeat - until*
- Вложенные циклы
- *Массивы, списки, множества*
- Процедуры и функции
- Операторы *exit, continue, break*
- Операции ввода и вывода *Read(ln), Write(ln).*

Для кого эта книга

Занимательные задачи можно с успехом использовать на уроках информатики и математики – как эффективное средство для укрепления навыков решения задач и программирования. **Учителям информатики и математики** эта книга может быть полезна и при подготовке интегрированных уроков.

Решение занимательных задач интересно и само по себе, поэтому я надеюсь, что книга заинтересует **любителей различного рода задач и головоломок**.

Так как в книге предложен способ решения задач на компьютере, то **начинающие программисты и любители программирования** найдут в ней немало полезных упражнений и советов для себя.

И наконец, **родители**, которые хотели бы привлечь детей к программированию и тем самым развить их логическое, алгоритмическое мышление, должны обратить самое пристальное внимание на эту книгу.

Валерий Рубанцев

Условные обозначения, принятые в книге:

Дополнение или замечание

Требование или указание

Исходный код

Задание для самостоятельного решения

Заголовок проекта:

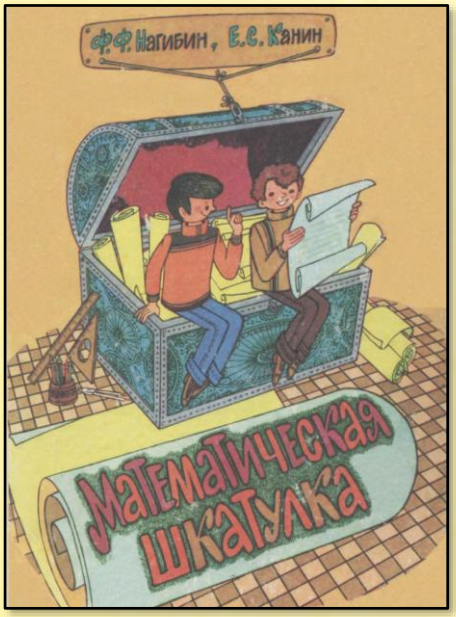
Проект

Исходные коды всех проектов находятся в папке **_Projects**

Оглавление

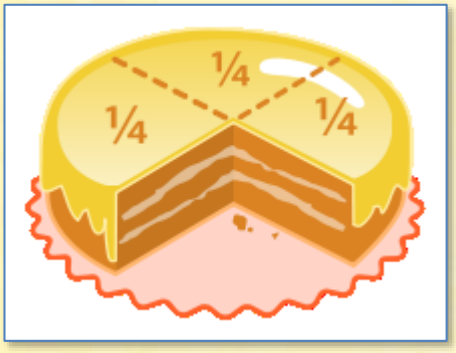
Задачи Юрия Нагибина

Одна из самых известных книг по занимательной математике - **Математическая шкатулка** Фёдора Фёдоровича Нагибина - была 4 раза издана на русском языке, а также переведена на на украинский и японский языки.



Дроби

Задача 267 из книги *Математическая шкатулка* [Нагибин88], страница 43:



> Сравните дроби:
>
> $$\frac{23}{99}, \frac{2323}{9999}, \frac{232323}{999999}$$

Проще всего сравнить дроби, найдя их **десятичные** значения.

> Если смекнуть, то задача легко решается и в уме.

Для этого следует разделить числитель на знаменатель при помощи оператора деления /. Важно учесть, что в этой задаче все числители и знаменатели – целые числа, а результаты операций – **действительные** числа.

Для удобства операцию деления мы выполним в функции *Solve*, которая получает 2 целых числа, а возвращает частное от деления первого числа на второе.

```
uses
   System;

// Нагибин 267


begin
   // заголовок окна:
   Console.Title := 'Нагибин 267';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Дроби');
   Console.ForegroundColor := ConsoleColor.Yellow;
   Console.WriteLine();
```

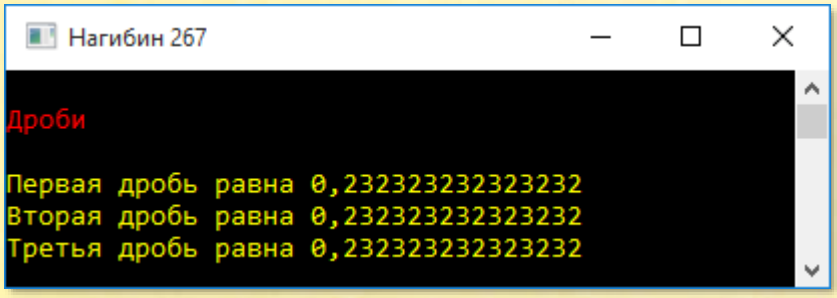
```
    var res := Solve(23, 99);
    Console.WriteLine('Первая дробь равна {0}', res);

    res := Solve(2323, 9999);
    Console.WriteLine('Вторая дробь равна {0}', res);

    res := Solve(232323, 999999);
    Console.WriteLine('Третья дробь равна {0}', res);


    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```

Тип функции Solve должен быть *double* (или *decimal*).

Чтобы при делении целых чисел получить правильное, дробное значение нужно первое число привести к типу *double*:

```
//РЕШАЕМ ЗАДАЧУ
function Solve(n1, n2: integer): double := double(n1) / n2;
```

На рисунке вы видите, что все три дроби равны!



Дроби, скобки и квадраты

Задача 482 из книги *Математическая шкатулка* **[Нагибин88]**, страница 88:

Найдите простой приём вычисления:

1) $\left(2\frac{1}{3}+\frac{1}{9}\right)\cdot 9$

2) $\left(3\frac{2}{5}+\frac{4}{7}\right)\cdot 70$

3) $7\frac{1}{2}\cdot 6\frac{1}{2}$

4) $11\frac{3}{4}\cdot 12\frac{1}{4}$

5) 98^2-4

6) $\frac{17^2-15^2}{32}$

7) $\frac{106}{47}-\frac{94}{53}$

Здесь мы встречаемся с дробями, у которых имеется целая часть, поэтому каждую дробь нужно записывать как **сумму** целой и дробной частей.

В дробной части числитель (или знаменатель) должен иметь тип *double*. Поскольку все числители – целые литералы, то достаточно поставить после числа точку и нуль, чтобы превратить их в действительные литералы типа *double*.

Знак умножения * нужно ставить всегда, хотя в математических формулах он часто опускается.

Операцию возведения в квадрат заменяйте умножением.

Не забывайте ставить круглые скобки!

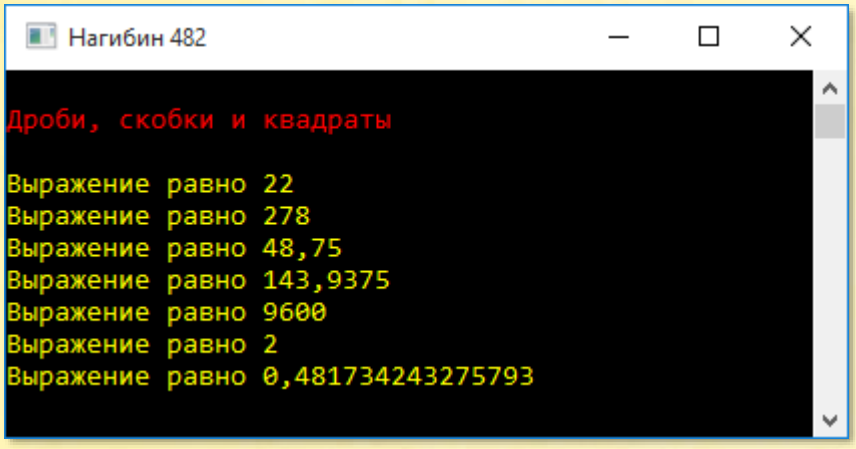
С учётом этих замечаний вы легко перепишите условия задачи:

```
uses
   System;
```

```
// Нагибин 482


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
    Console.ForegroundColor := ConsoleColor.Yellow;

    var res := (2 + 1.0 / 3 + 1.0 / 9) * 9;
    Console.WriteLine('Выражение равно {0}', res);

    res := (3 + 2.0 / 5 + 4.0 / 7) * 70;
    Console.WriteLine('Выражение равно {0}', res);

    res := (7 + 1.0 / 2) * (6 + 1.0 / 2);
    Console.WriteLine('Выражение равно {0}', res);

    res := (11 + 3.0 / 4) * (12 + 1.0 / 4);
    Console.WriteLine('Выражение равно {0}', res);

    res := 98 * 98 - 4;
    Console.WriteLine('Выражение равно {0}', res);

    res := (17 * 17 - 15 * 15) / 32.0;
    Console.WriteLine('Выражение равно {0}', res);

    res := 106.0 / 47 - 94.0 / 53;
    Console.WriteLine('Выражение равно {0}', res);

    Console.WriteLine();
end;


begin
   // заголовок окна:
   Console.Title := 'Нагибин 482';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Дроби, скобки и квадраты');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
```

```
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

И ответ готов:



Степенные выражения

Задача 483 из книги *Математическая шкатулка* [Нагибин88], страница 88:

Найдите числовое значение выражения возможно быстрее:

1) $(a+b)^2-(a-b)^2$ при $a=\frac{17}{37}$, $b=-2\frac{3}{17}$;

2) $a^2(a+b^2)(a^4-b^{10})(a^2-b)$ при $a=5$ и $b=25$;

3) $\frac{m^2(m+n^2)(m^3-n^6)(m^2-n)}{m^2+n^2}$ при $m=4$ и $n=16$;

4) $x^2-86x+113$ при $x=87$.

В этой задаче мы имеем дело не только с числовыми литералами, но и с настоящими **переменными** *a, b, m, n* и *x*. Легко заметить, что в первой подзадаче переменные имеют тип *double*, а во всех остальных - тип *integer*.

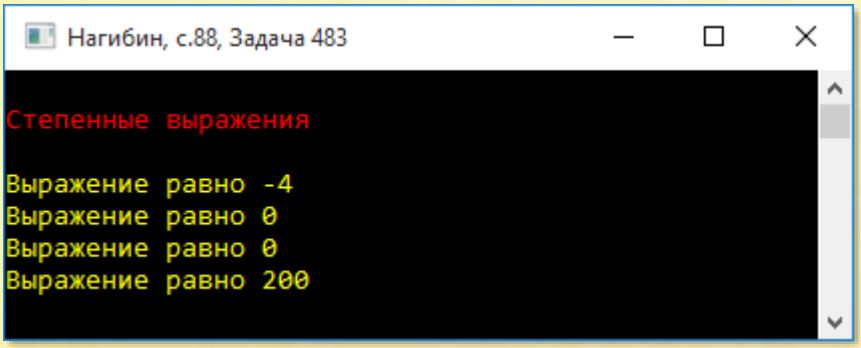
Тип выражения также определить нетрудно: первое и третье выражения имеют тип *double*, второе и четвёртое - *integer*.

Степени переменных лучше заменить произведением.

```
uses
   System;

// Нагибин 483


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
    Console.ForegroundColor := ConsoleColor.Yellow;

    var a1 := 17.0 / 37;
    var b1 := -(2 + 3.0 / 17);
    var ab1 := a1 + b1;
    var ab2 := a1 - b1;
    var res1 := ab1 * ab1 - ab2 * ab2;
    Console.WriteLine('Выражение равно {0}', res1);

    var a := 5;
    var b := 25;
    var res := (a * a) *
               (a + b * b) *
               (a*a*a*a - b*b*b*b*b*b*b*b*b*b) *
               (a * a - b);
    Console.WriteLine('Выражение равно {0}', res);

    var m := 4;
    var n := 16;
    res1 := m * m * (m + n * n) * (m * m * m -
            n*n*n*n*n*n) *
           (m * m - n) / (m * m + n * n);
    Console.WriteLine('Выражение равно {0}', res1);
```

```
    var x := 87;
    res := x * x - 86 * x + 113;
    Console.WriteLine('Выражение равно {0}', res);

    Console.WriteLine();
end;


begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.88, Задача 483';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Степенные выражения');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Если вы нигде не ошиблись при переводе формул, то должны получить такие ответы:



Переменное выражение

Задача 484 из книги *Математическая шкатулка* **[Нагибин88]**, страница 88:

Найдите числовое значение выражения $\frac{a^2b+ab^2}{ab+b^2}$ при:

1) $a=2\frac{1}{2},\ b=\frac{3}{7}$

2) $a=15{,}2,\ b=12{,}3$

3) $a=-5,\ b=5$

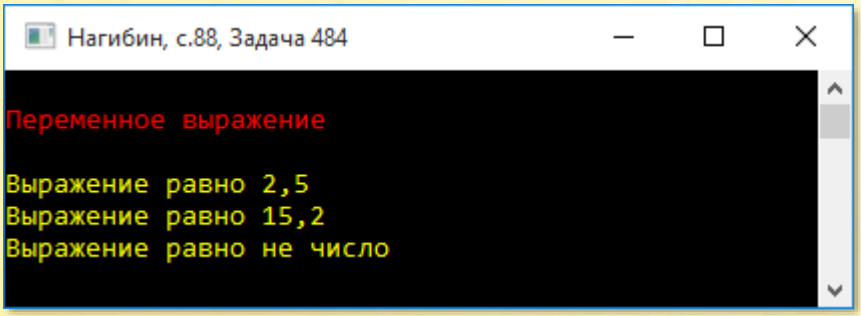
Поскольку в этой задаче нужно найти 3 значения одного выражения, то его следует вычислять в отдельной функции *Solve*, которая получает значения переменных *a* и *b*, а возвращает результат выполнения всех действий над ними.

Совершенно очевидно, что параметры функции *Solve* и возвращаемое значение должны иметь тип **double**, так как значения переменных в задаче – дробные числа:

```
uses
   System;

// Нагибин 484

// РЕШАЕМ ЗАДАЧУ
function Solve(a, b: double): double;
begin
   Result := (a * a * b + a * b * b) / (a * b + b * b);
end;

begin
```

```
    // заголовок окна:
    Console.Title := 'Нагибин, с.88, Задача 484';
    Console.WriteLine('');
    Console.ForegroundColor := ConsoleColor.Red;
    Console.WriteLine('Переменное выражение');
    Console.WriteLine();

    Console.ForegroundColor := ConsoleColor.Yellow;
    var res := Solve(2 + 1.0 / 2, 3.0 / 7);
    Console.WriteLine('Выражение равно {0}', res);

    res := Solve(15.2, 12.3);
    Console.WriteLine('Выражение равно {0}', res);

    res := Solve(-5, 5);
    Console.WriteLine('Выражение равно {0}', res);


    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```

Третья часть задачи – с подвохом: при значениях *a = -5, b = 5* знаменатель дроби обращается в **нуль**, а на нуль делить нельзя. На рисунке это обстоятельство отмечено - результат вычисления не является числом:



Корни

Задача 487 из книги *Математическая шкатулка* [Нагибин88], страница 89:

Найдите наиболее простой способ вычисления:

1) $\sqrt{34 \cdot 8 \cdot 17}$

2) $\sqrt{39 \cdot 27 \cdot 52}$

3) $\sqrt{22 \cdot 11 \cdot 54 \cdot 48}$

4) $\sqrt{82^2 - 18^2}$

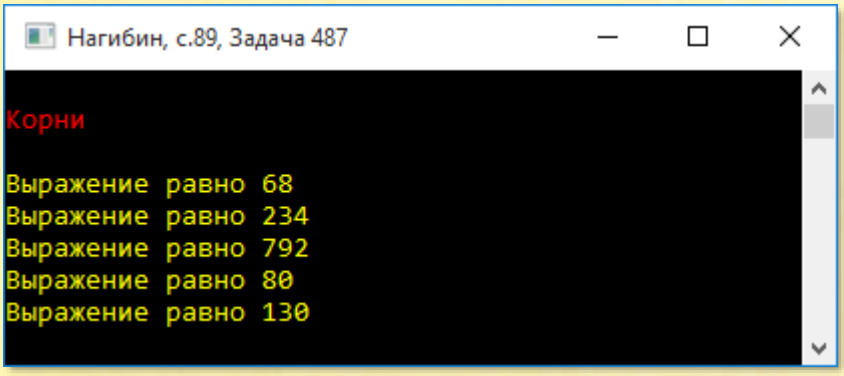
5) $\sqrt{(134{,}5)^2 - (34{,}5)^2}$

Для вычисления квадратных корней используйте функцию Sqrt:

```
uses
   System;

// Нагибин 487


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
    Console.ForegroundColor := ConsoleColor.Yellow;

    var res := Sqrt(34 * 8 * 17);
    Console.WriteLine('Выражение равно {0}', res);

    res := Sqrt(39 * 27 * 52);
    Console.WriteLine('Выражение равно {0}', res);

    res := Sqrt(22 * 11 * 54 * 48);
    Console.WriteLine('Выражение равно {0}', res);
```

```
    res := Sqrt(82 * 82 - 18 * 18);
    Console.WriteLine('Выражение равно {0}', res);

    res := Sqrt(134.5 * 134.5 - 34.5 * 34.5);
    Console.WriteLine('Выражение равно {0}', res);

    Console.WriteLine();
end;


begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.89, Задача 487';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Корни');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Несмотря на некруглые числа, ответы мы получили изумительно круглые:



Чтобы извлечь кубический корень, не обойтись без функции Power. При этом нужно учитывать, что извлечение кубического корня равносильно возведению числа в степень 1.0/3.

В книге Кордемского и Ахадова *Удивительный мир чисел* [КА86], на странице 42 приводится пара интересных пятизначных чисел:

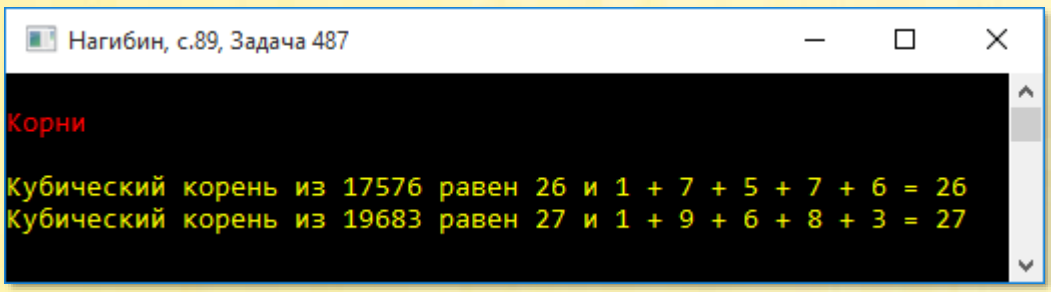
$$1+7+5+7+6=\sqrt[3]{17576}$$
$$1+9+6+8+3=\sqrt[3]{19683}$$

Кубические корни из этих чисел в точности равны сумме их цифр!

Давайте проверим это утверждение, а заодно и кубическими корнями побалуемся, как говорил Карлсон:

```
procedure Solve2();
  begin
      Console.ForegroundColor := ConsoleColor.Yellow;

      var res := Power(17576, 1.0 / 3);
      Console.WriteLine('Кубический корень из 17576 равен {0} и
                         1 + 7 + 5 + 7 + 6 = {1}',
                         res, 1 + 7 + 5 + 7 + 6);

      res := Power(19683, 1.0 / 3);
      Console.WriteLine('Кубический корень из 19683 равен {0} и
                         1 + 9 + 6 + 8 + 3 = {1}',
                         res, 1 + 9 + 6 + 8 + 3);
  end;
```

Рисунок убеждает нас в правоте авторов книги:



Четыре числа

Задача 341 из книги *Математическая шкатулка* [Нагибин88]:

> Произведение четырёх последовательных натуральных чисел равно 3024.
>
> Найдите эти числа.

Простенькая задача на 1 бесконечный цикл *for*:

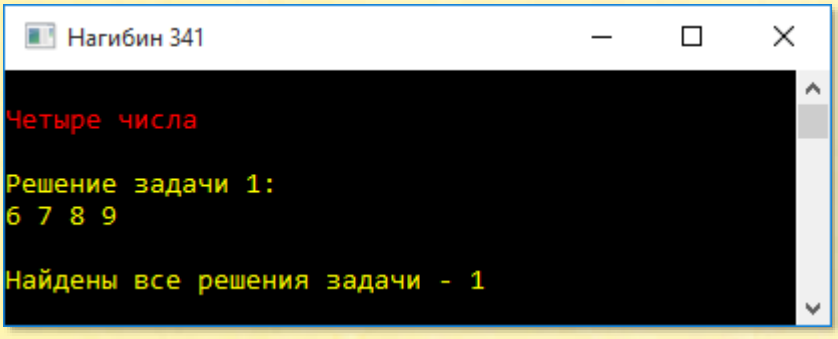
```
uses
   System;

// Нагибин 341

begin
   // заголовок окна:
   Console.Title := 'Нагибин 341';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Четыре числа');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   for var i := 1 to integer.MaxValue do
   begin
```

```
        //произведение четырёх чисел:
        var mult := i * (i + 1) * (i + 2) * (i + 3);
```

Здесь, как обычно, нужно не забыть об условии прекращения цикла. В этой задаче цикл можно закончить, когда произведение чисел превысит заданное значение:

```
        if (mult > 3024) then
        begin
           Console.WriteLine('Найдены все решения задачи - ' + nVar);
           exit;
        end
        else if (mult = 3024) then
        begin
           nVar += 1;
           Console.WriteLine('Решение задачи {0}: ', nVar);
           Console.WriteLine('{0} {1} {2} {3}', i, (i + 1), (i + 2),
                             (i + 3));
           Console.WriteLine();
        end;
    end
end;
```

На рисунке вы видите, что в произведении участвуют числа 6, 7, 8 и 9:



Пятизначное число

Задача 440 из книги *Математическая шкатулка* [Нагибин88]:

> Мне было задано пятизначное число. К этому числу нужно было прибавить 200 000 и сумму умножить на 3. Вместо этого я приписал к заданному мне числу в конце его справа цифру 2 и получил правильный результат.
>
> Какое число было мне задано?

Пятизначных чисел совсем немного, поэтому мы легко переберём их в цикле *for*:

```
uses
   System;

// Нагибин 440

begin
   // заголовок окна:
   Console.Title := 'Нагибин 440';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Пятизначное число');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   for var i := 10000 to 99999 do
```

```
    begin
```

По условию задачи, к заданному пятизначному числу *i* следовало прибавить 200 000, а получившуюся сумму – умножить на 3:

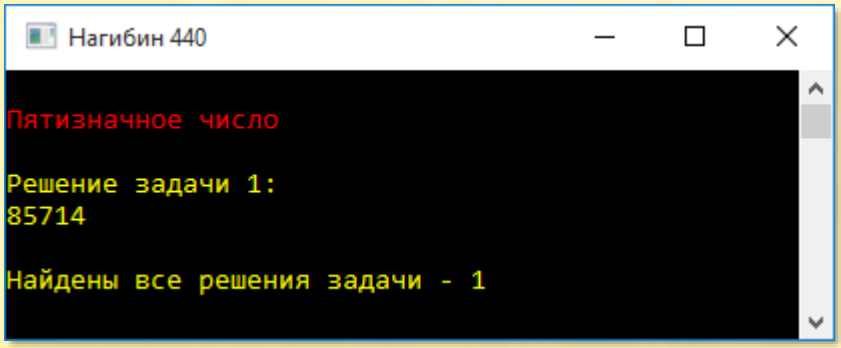
```
var num1 := (i + 200000) * 3;
```

Вместо этого нерадивый ученик приписал в конце числа *i* двойку. Для этого мы должны умножить заданное число на 10 и добавить 2:

```
var num2 := i * 10 + 2;
```

Когда эти два числа совпадут, задача будет решена:

```
      if (num1 = num2) then
      begin
         nVar += 1;
         Console.WriteLine('Решение задачи {0}: ', nVar);
         Console.WriteLine('{0}', i);
         Console.WriteLine();
      end
   end;
   Console.WriteLine('Найдены все решения задачи - ' + nVar);
   Console.WriteLine();
end;
```



На всякий случай мы на этом не останавливаемся и продолжаем проверять пятизначные числа до полного их исчерпания. Впрочем, как показывает рисунок, задача имеет *единственное* решение, а искомое число – 85714.

Половина и треть

Задача 489 из книги *Математическая шкатулка* [Нагибин88]:

> Запишите наименьшее натуральное число, половина и треть которого были бы соответственно квадратом и кубом некоторых натуральных чисел.

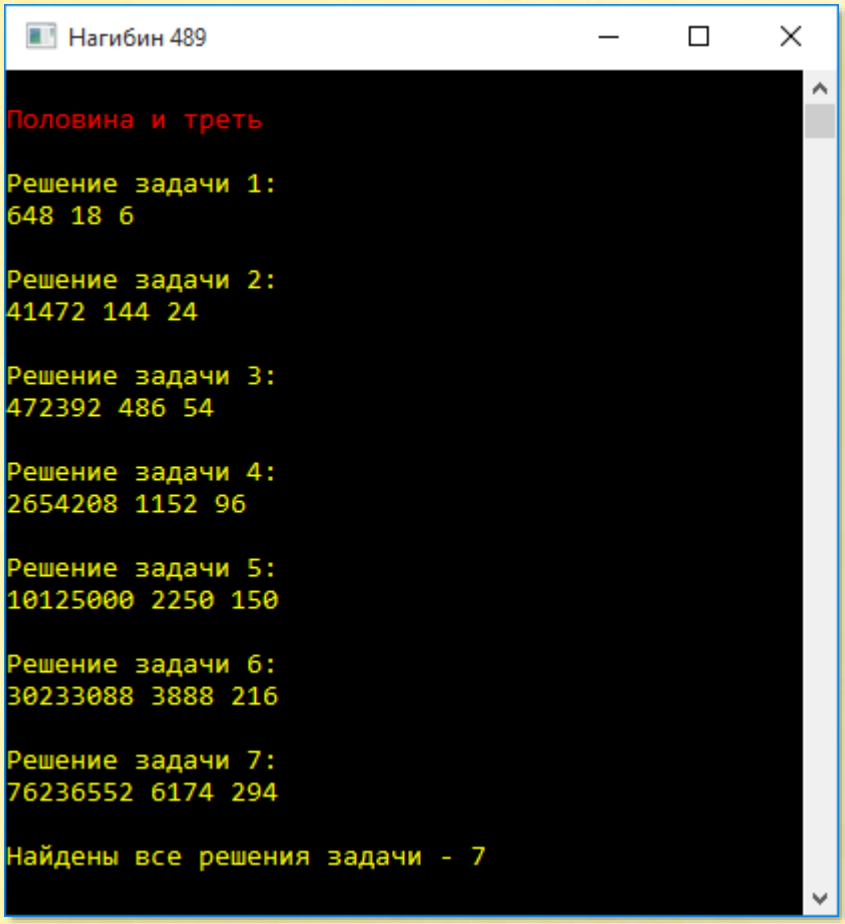
Так как искомое число одновременно делится на *2* и на *3*, то его следует искать среди чисел, кратных 6.

```
uses
   System;

// Нагибин 489


begin
   // заголовок окна:
   Console.Title := 'Нагибин 489';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Половина и треть');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Мы не ограничимся поиском единственного числа с означенными свойствами, а проверим все числа до 100 миллионов:

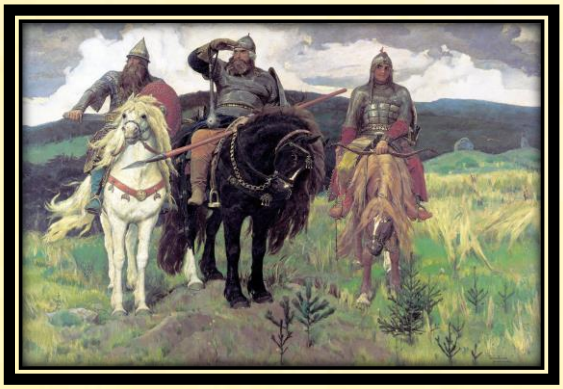
```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   var start := 6;
   var step := 6;
   var i := start;
   while(i < 100000000) do
   begin
      var num2 := i div 2;
      var num3 := i div 3;
      if (IsQuadrat(num2) and IsQube(num3)) then
      begin
         nVar += 1;
         Console.WriteLine('Решение задачи {0}: ', nVar);
         Console.WriteLine('{0} {1} {2}', i,
                           Math.Sqrt(num2),
                           Math.Pow(num3, 1.0 / 3.0));
         Console.WriteLine();
      end;
      i += step;
   end;
   Console.WriteLine('Найдены все решения задачи - ' + nVar);
   Console.WriteLine();
end;
```

Задача – совершенно компьютерная, поэтому он справляется с ней в считанные мгновения. Среди первых 100 миллионов чисел только 7 удовлетворяют условиям задачи, а наименьшее из них – 648:



Три числа

Задача 554 из книги *Математическая шкатулка* [Нагибин88]:



> Из двузначного числа, умноженного на однозначное, вычли однозначное и получили 1.
>
> Какие это были числа?

Обозначим двузначное число буквой ***x***, а однозначные - буквами ***y*** и ***z***. Так как нам нужно найти три числа, то решить задачу можно с помощью *трёх* вложенных циклов ***for***. Но давайте обойдёмся **двумя**.

По условию задачи,

$$x * y - z = 1 \qquad (1)$$

Или

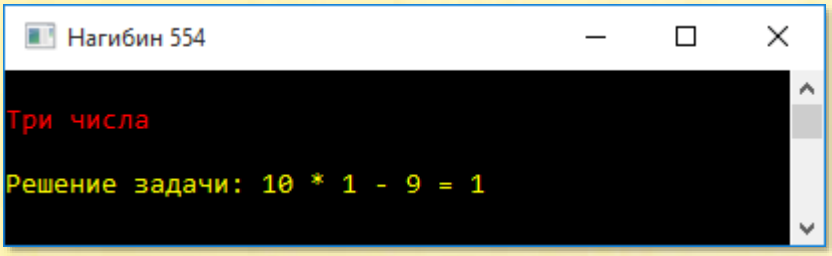
$$z = x * y - 1 \qquad (2)$$

Теперь достаточно в двух циклах *for* изменять значения переменных *x* и *y*. Значение переменной *z* мы находим их уравнения (2). Если это **однозначное** число, то задача решена!

```
uses
   System;

// Нагибин 554


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   for var x := 10 to 99 do
      for var y := 1 to 9 do
      begin
         var z := x * y - 1;
         if ((z > 0) and (z < 10)) then
         begin
            Console.Write('Решение задачи: ');
            Console.WriteLine('{0} * {1} - {2} = 1', x, y, z);
            Console.WriteLine();
         end
      end
end;


begin
```

```
    // заголовок окна:
    Console.Title := 'Нагибин 554';
    Console.WriteLine('');
    Console.ForegroundColor := ConsoleColor.Red;
    Console.WriteLine('Три числа');
    Console.ForegroundColor := ConsoleColor.Green;
    Console.WriteLine();

    Solve();

    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```

Как и следовало ожидать, задача имеет *единственное* и очень простое решение:



Пятизначный куб

Задача 556 из книги *Математическая шкатулка* [Нагибин88]:

> Задумано пятизначное число, являющееся кубом натурального числа.
>
> Восстановите задуманное число, если известно, что оно должно делиться на 3 и последняя цифра его 6.

Мы не знаем, какое натуральное число задумано, поэтому будем перебирать все натуральные числа, начиная с единицы:

```
uses
   System;

// Нагибин 556

begin
   // заголовок окна:
   Console.Title := 'Нагибин 556';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Пятизначный куб');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

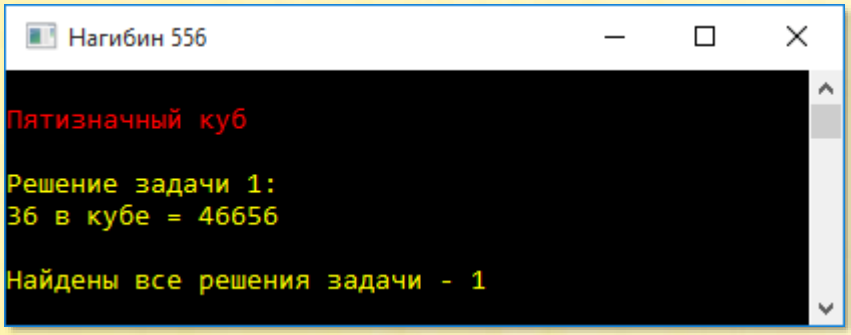
Числа, которые короче пятизначных, мы пропускаем, а бесконечный цикл *for* заканчиваем сразу, как только куб станет шестизначным:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;

   for var i := 1 to integer.MaxValue do
   begin
      var qube := i * i * i;
      // ищем только пятизначные числа:
      if (qube > 99999) then
         break;
      if (qube < 10000) then
         continue;
```

Задуманное пятизначное число должно быть кратно трём и заканчиваться на шестёрку:

```
      if (qube mod 3 <> 0) then
          continue;
       if (qube mod 10 = 6) then
       begin
          nVar += 1;
          Console.WriteLine('Решение задачи {0}: ', nVar);
          Console.WriteLine('{0} в кубе = {1}', i, qube);
          Console.WriteLine();
       end
    end;
    Console.WriteLine('Найдены все решения задачи - ' + nVar);
    Console.WriteLine();
end;
```

Рисунок показывает, что задача имеет *единственное* решение, а задумано было число 46 656 (в книге указан неверный ответ – 45 656):



Разность кубов

Задача 561 из книги *Математическая шкатулка* [Нагибин88]:

> Разность кубов двух последовательных натуральных чисел равна 331.
>
> Восстановите эти числа.

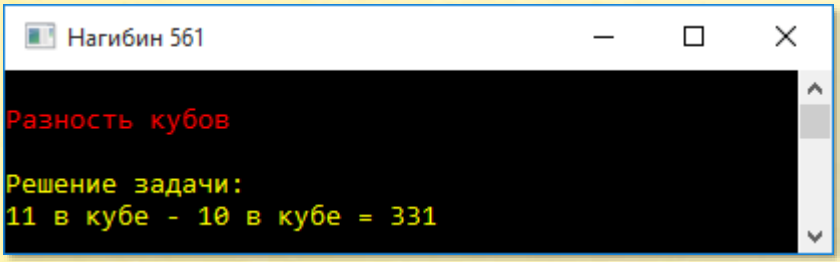
$$x^2 - y^2 = (x + y)(x - y)$$
$$(x \pm y)^2 = x^2 \pm 2xy + y^2$$
$$x^3 + y^3 = (x + y)(x^2 - xy + y^2)$$
$$x^3 - y^3 = (x - y)(x^2 + xy + y^2)$$

В бесконечном цикле *for* перебираем натуральные числа – до тех пор, пока не найдём пару чисел, удовлетворяющих условию задачи:

```
uses
   System;

// Нагибин 561


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   for var i := 1 to integer.MaxValue do
   begin
      var qube1 := i * i * i;
      var j := i + 1;
      var qube2 := j * j * j;
      if (qube2 - qube1 = 331) then
      begin
         Console.WriteLine('Решение задачи: ');
         Console.WriteLine('{1} в кубе - {0} в кубе = 331',
                                    i, j);
         break;
      end
   end;
   Console.WriteLine();
```

```
end;


begin
   // заголовок окна:
   Console.Title := 'Нагибин 561';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Разность кубов');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Простой ответ на простую задачу:



Двузначное число

Задача 563 из книги *Математическая шкатулка* [Нагибин88]:

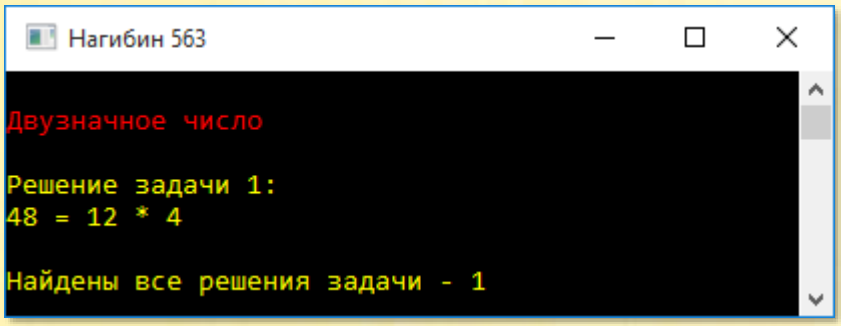
> Найдите двузначное положительное число, равное произведению суммы и разности его цифр.

Единственная «сложность» в решении этой задачи – проследить, чтобы разность цифр была неотрицательной:

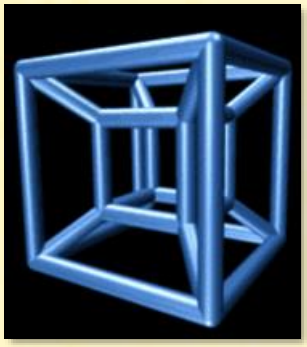
```
uses
   System;

// Нагибин 563


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   for var i := 10 to 100 - 1 do
   begin
      var sum := i div 10 + i mod 10;
      var sub := i div 10 - i mod 10;
      if (sub < 0) then
         sub *= -1;
      if (i = sum * sub) then
      begin
         nVar += 1;
         Console.WriteLine('Решение задачи {0}: ', nVar);
         Console.WriteLine('{0} = {1} * {2}', i, sum, sub);
         Console.WriteLine();
      end
   end;
```

```
    Console.WriteLine('Найдены все решения задачи - ' + nVar);
    Console.WriteLine();
end;

begin
    // заголовок окна:
    Console.Title := 'Нагибин 563';
    Console.WriteLine('');
    Console.ForegroundColor := ConsoleColor.Red;
    Console.WriteLine('Двузначное число');
    Console.ForegroundColor := ConsoleColor.Green;
    Console.WriteLine();

    Solve();

    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```

Двузначное число равно 48. Сумма его цифр - 12, а разность - четыре. Рисунок показывает, что задача решена верно:



Квадратный куб



Задача 584 из книги *Математическая шкатулка* [Нагибин88]:

> Какое трёхзначное число равно кубу цифр его единиц, а также квадрату числа, составленного из второй и первой цифр?

Перебираем в цикле *for* все трёхзначные числа:

```
uses
   System;

// Нагибин 584

begin
   // заголовок окна:
   Console.Title := 'Нагибин 584';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Квадратный куб');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;

   for var i := 100 to 1000 do
   begin
```

Последняя цифра числа равна остатку от его деления на 10:

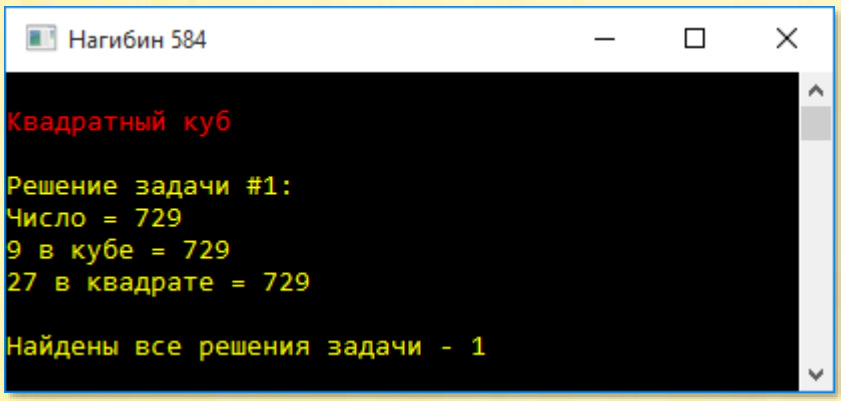
```
var ed := i mod 10;
var qube := ed * ed * ed;
```

Немного труднее переставить местами первую и вторую цифры числа:

```
var n21 := (i div 10 mod 10) * 10 + i div 100;
var quad := n21 * n21;
```

Если очередное трёхзначное число *i* выдержало проверки, то мы печатаем ответ на экране:

```
      if ((i = qube) and (i = quad)) then
      begin
         nVar += 1;
         Console.WriteLine('Решение задачи #{0}: ', nVar);
         Console.WriteLine('Число = {0}', i);
         Console.WriteLine('{0} в кубе = {1}', ed, qube);
         Console.WriteLine('{0} в квадрате = {1}', n21, quad);

         Console.WriteLine();
      end
   end;
   Console.WriteLine('Найдены все решения задачи - ' + nVar);
   Console.WriteLine();
end;
```



Задача имеет *единственное* решение:

$729 = 9^3 = 27^2$

Квадрат и куб

Задача 604 из книги *Математическая шкатулка* [Нагибин88]:

> Найдите два двузначных числа, куб одного из которых равен квадрату другого.

Когда перебор вариантов, как в этой задаче, небольшой, то вполне разумно прибегнуть к методу грубой силы, хотя задачу можно решить и куда изящнее!

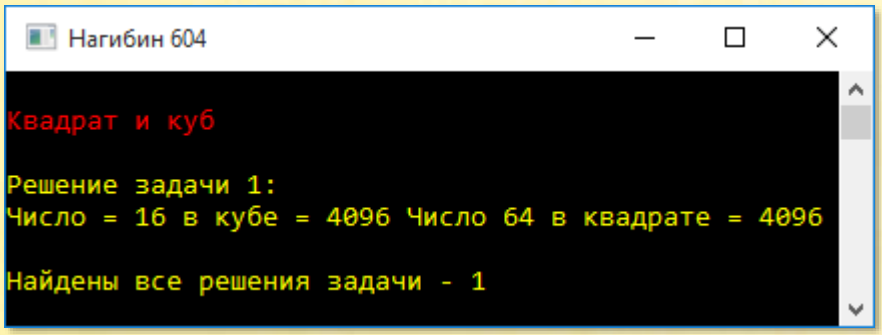
```
uses
   System;

// Нагибин 604

begin
   // заголовок окна:
   Console.Title := 'Нагибин 604';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Квадрат и куб');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Итак, в двух вложенных циклах ***for*** мы перебираем все двузначные числа, учитывая, что число, которое возводят в куб (*i*), очевидно меньше числа, которое возводят в квадрат (*j*):

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
```

```
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   for var i := 10 to 99 do
   begin
      for var j := i + 1 to 100 - 1 do
      begin
         if (i * i * i = j * j) then
         begin
            nVar += 1;
            Console.WriteLine('Решение задачи {0}: ', nVar);
            Console.WriteLine('Число = {0} в кубе = {1}
                               Число {2} в квадрате = {3}',
                               i, i * i * i, j, j * j);
            Console.WriteLine();
         end
      end
   end;
   Console.WriteLine('Найдены все решения задачи - ' + nVar);
   Console.WriteLine();
end;
```

Через пару минут весь исходный код написан – и задача решена:



Четырёхзначное число

Задача 620 из книги *Математическая шкатулка* [Нагибин88]:

> **Найдите нечётное четырёхзначное число**, две средние цифры которого образуют число в пять раз больше числа тысяч и втрое больше числа единиц этого числа.

Весь «фокус» этой задачи заключается в том, чтобы правильно **извлечь цифры** из числа.

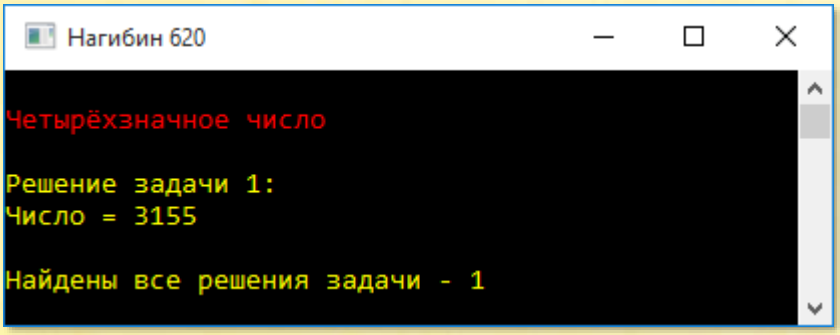
```
uses
   System;

// Нагибин 620

begin
   // заголовок окна:
   Console.Title := 'Нагибин 620';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Четырёхзначное число');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Так как мы не можем быть уверены, что задача имеет единственное решение, то просто перебираем в цикле ***for*** все *нечётные* четырёхзначные числа:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
```

```
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   // четырёхзначное число:
   var i := 1001;
   while(i <= 9999) do
   begin
      // две средние цифры:
      var n23 := i div 10 mod 100;
      // число тысяч:
      var n1 := i div 1000;
      // число единиц:
      var n4 := i mod 10;

      if ((n23 = n1 * 5) and (n23 = n4 * 3)) then
      begin
         nVar += 1;
         Console.WriteLine('Решение задачи {0}: ', nVar);
         Console.WriteLine('Число = {0}', i);
         Console.WriteLine();
      end;
      i += 2;
   end;
   Console.WriteLine('Найдены все решения задачи - ' + nVar);
   Console.WriteLine();
end;
```

Решение задачи показано на рисунке:



Трёхзначное число

Задача 622 из книги *Математическая шкатулка* [Нагибин88]:

> Трёхзначное число, две первые цифры которого одинаковы, а цифра единиц 5, разделили на однозначное и получили 8.
>
> Найдите делимое, делитель и частное.

Нам понадобятся 2 цикла *for*: первый - чтобы составить трёхзначное число:

```
uses
   System;

// Нагибин 622

begin
   // заголовок окна:
   Console.Title := 'Нагибин 622';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Трёхзначное число');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   for var i := 1 to 10 - 1 do
   begin
      //трёхзначное число:
```

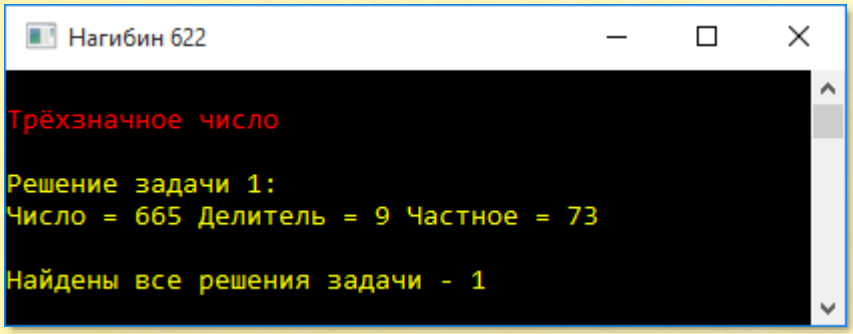
```
        var num := i * 100 + i * 10 + 5;
```

Второй – чтобы подобрать к нему все однозначные делители:

```
for var j := 2 to 10 - 1 do
begin
```

Когда остаток от деления трёхзначного числа num на однозначное *j* станет равным восьми, мы печатаем ответ на экране:

```
            if (num mod j = 8) then
            begin
               nVar += 1;
               Console.WriteLine('Решение задачи {0}: ', nVar);
               Console.WriteLine('Число = {0} Делитель = {1}
                                  Частное = {2}', num, j, num div j);
               Console.WriteLine();
            end
         end
      end;
      Console.WriteLine('Найдены все решения задачи - ' + nVar);
      Console.WriteLine();
end;
```



Зачёркнутая цифра

Задача 600 из книги *Математическая шкатулка* [Нагибин88]:

> В трёхзначном числе зачеркнули цифру сотен, затем полученное двузначное число умножили на 7 и получили вновь исходное трёхзначное число.
>
> Какое это число?

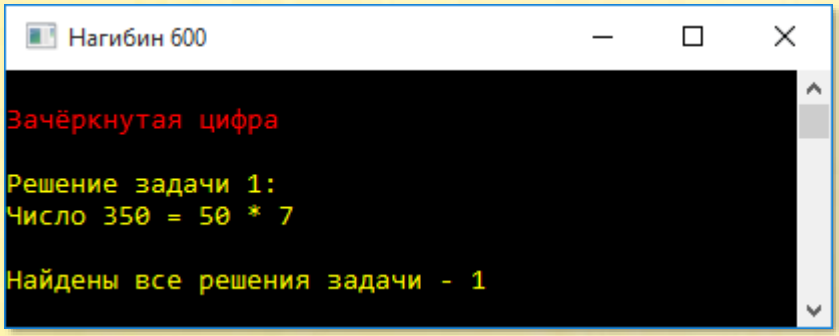
Для искушённого математическими задачами программиста решать такие примеры – что семечки щёлкать.

```
uses
   System;

// Нагибин 600

begin
   // заголовок окна:
   Console.Title := 'Нагибин 600';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Зачёркнутая цифра');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Только вместо реального зачёркивания цифры сотен нужно найти остаток от деления трёхзначного числа на сотню:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   for var i := 100 to 1000 - 1 do
   begin
      // двузначное число:
      var n2 := i mod 100;
      if (i = n2 * 7) then
      begin
         nVar += 1;
         Console.WriteLine('Решение задачи {0}: ', nVar);
         Console.WriteLine('Число {0} = {1} * 7', i, n2);
         Console.WriteLine();
      end
   end;
   Console.WriteLine('Найдены все решения задачи - ' + nVar);
   Console.WriteLine();
end;
```

А вот и ответ на задачу:



Зачёркнутая цифра 2

Задача 587 из книги *Математическая шкатулка* [Нагибин88]:

> В трёхзначном числе зачеркнули среднюю цифру. Полученное двузначное число оказалось в 6 раз меньше исходного трёхзначного.
>
> Какое такое трёхзначное число.

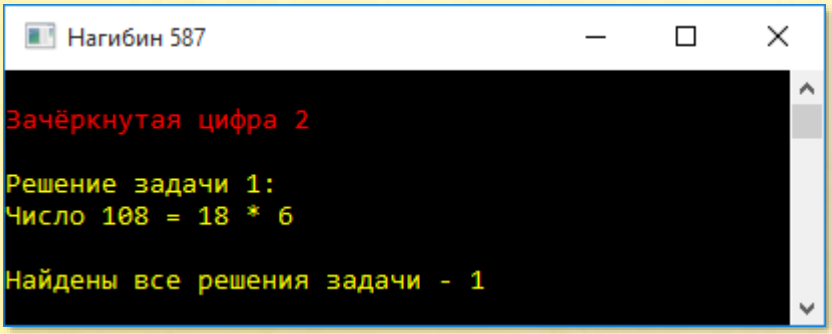
Можно щёлкать задачки, как семечки, а можно – как орешки!

```
uses
   System;

// Нагибин 587

begin
   // заголовок окна:
   Console.Title := 'Нагибин 587';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Зачёркнутая цифра 2');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Опять самая «сложная» часть задачи заключается в зачёркивании цифры. На этот раз – средней:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   for var i := 100 to 1000 - 1 do
   begin
      // двузначное число:
      var n2 := i div 100 * 10 + i mod 10;
      if (i = n2 * 6) then
      begin
         nVar += 1;
         Console.WriteLine('Решение задачи {0}: ', nVar);
         Console.WriteLine('Число {0} = {1} * 6', i, n2);
         Console.WriteLine();
      end
   end;
   Console.WriteLine('Найдены все решения задачи - ' + nVar);
   Console.WriteLine();
end;
```

Орешки, как и семечки, нам по зубам:



Трёхзначное число 2

Задача 585 из книги *Математическая шкатулка* [Нагибин88]:

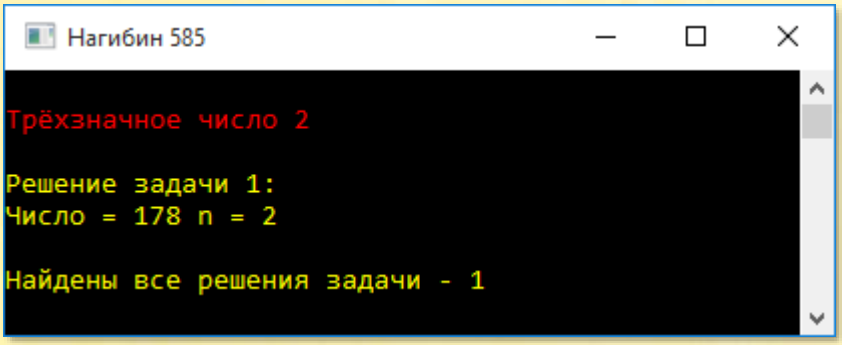
> Если первую цифру трёхзначного числа увеличить на *n*, а вторую и третью цифру уменьшить на *n*, то получится число в *n* раз больше исходного.
>
> Найдите *n* и исходное число.

А решение этой задачи и в комментариях не нуждается:

```
uses
   System;

// Нагибин 585


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nVar := 0;
   for var i := 100 to 1000 - 1 do
   begin
      // первая цифра:
      var n1 := i div 100;
      // вторая цифра::
      var n2 := i div 10 mod 10;
      // третья цифра::
      var n3 := i mod 10;
      for var n := 1 to 9 do
      begin
         if ((n1 + n) * 100 + (n2 - n) * 10 + (n3 - n) = i * n)
         then
         begin
            nVar += 1;
            Console.WriteLine('Решение задачи {0}: ', nVar);
```

```
            Console.WriteLine('Число = {0} n = {1}', i, n);
            Console.WriteLine();
          end
       end
    end;
    Console.WriteLine('Найдены все решения задачи - ' + nVar);
    Console.WriteLine();
end;


begin
    // заголовок окна:
    Console.Title := 'Нагибин 585';
    Console.WriteLine('');
    Console.ForegroundColor := ConsoleColor.Red;
    Console.WriteLine('Трёхзначное число 2');
    Console.ForegroundColor := ConsoleColor.Green;
    Console.WriteLine();

    Solve();

    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```

Трёхзначное число - 178, а однозначное – 2:



Последняя цифра

Задача 612 из книги *Математическая шкатулка* [Нагибин88]:

> Какая цифра стоит в конце числа, выражающего произведение:
>
> 9 • 11 • 13 • 15 • 17 • 19 • 21?

Было бы весьма неосторожно вычислять *полное* произведение, чтобы посмотреть на его последнюю цифру.

```
uses
   System;

// Нагибин 612

begin
   // заголовок окна:
   Console.Title := 'Нагибин 612';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Последняя цифра');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Достаточно в переменной *last* оставлять только последнюю цифру текущего произведения:

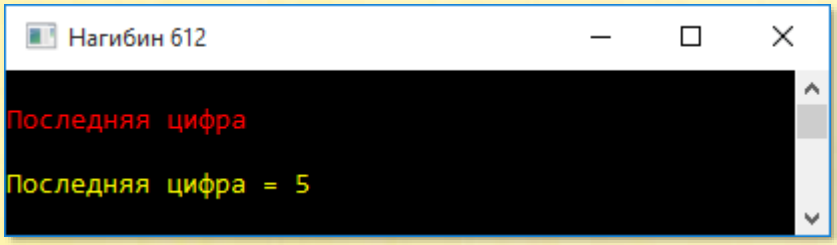
```
// РЕШАЕМ ЗАДАЧУ
```

```
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   // последняя цифра:
   var last := 1;
   var i := 9;
   while(i <= 21) do
   begin
      last := last * i mod 10;
      i += 2;
   end;
   Console.WriteLine('Последняя цифра = {0}', last);
   Console.WriteLine();
end;
```

Аналогично можно найти последнюю цифру ряда нечётных чисел любой длины!

Подумайте, изменится ли решение задачи, если нужно узнать последнюю цифру произведения **чётных** чисел!

А ответ на задачу вы можете видеть на рисунке:



Ящики

Процедура без параметров
Константы
Вложенные циклы for
Условный оператор if
Оператор continue
Форматированный вывод

Задача 424 из книги *Математическая шкатулка* [Нагибин88], страница 81:

Завод должен переслать заказчику 1100 деталей, которые для пересылки упаковываются в ящики трёх типов. Один ящик **первого** типа вмещает 70 деталей, **второго** типа – 40 деталей, **третьего** типа – 25 деталей. **Стоимость** пересылки одного ящика первого, второго и третьего типов равна соответственно 20, 10 и 7 р.

Какие ящики должен использовать завод, чтобы стоимость пересылки бала наименьшей? Недогрузка ящиков не допускается.

Эта типичная задача *линейного* (точнее - *целочисленного*) *программирования* может быть успешно решена простым перебором вариантов.

Примеры задач целочисленного программирования:

- задача о назначениях
- задача коммивояжера
- задача почтальона
- задача о максимальном паросочетании

```
uses
   System;
```

```
// Нагибин, с.81, Задача 424

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.81, Задача 424';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Ящики');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Обозначим через k1, k2 и k3 число ящиков каждого типа. Минимальное число этих ящиков равно 0. Верхнюю границу легко определить, поделив общее число деталей на вместимость каждого ящика:

```
procedure Solve();
const
   DETALI = 1100;
   CK1 = DETALI div 70;
   CK2 = DETALI div 40;
   CK3 = DETALI div 25;
```

Для хранения текущего значения минимальной стоимости пересылки мы заведём переменную minCost, которой сначала необходимо присвоить какое-либо большое значение:

```
begin
   // мин. стоимость пересылки:
   var minCost := Int32.MaxValue;
```

Во вложенных циклах *for* мы перебираем все варианты «расфасовки» деталей по ящикам, учитывая, что в сумме в них должно оказаться 1100 деталей:

```
   for var k1 := 0 to CK1 do
   begin
      for var k2 := 0 to CK2 do
      begin
         for var k3 := 0 to CK3 do
         begin
            if (k1 * 70 + k2 * 40 + k3 * 25 <> DETALI) then
               continue;
```

Для каждой удачной комбинации ящиков мы находим стоимость пересылки и, если она окажется меньше текущего значения переменной *minCost*, то печатаем текущую рекордную упаковку:

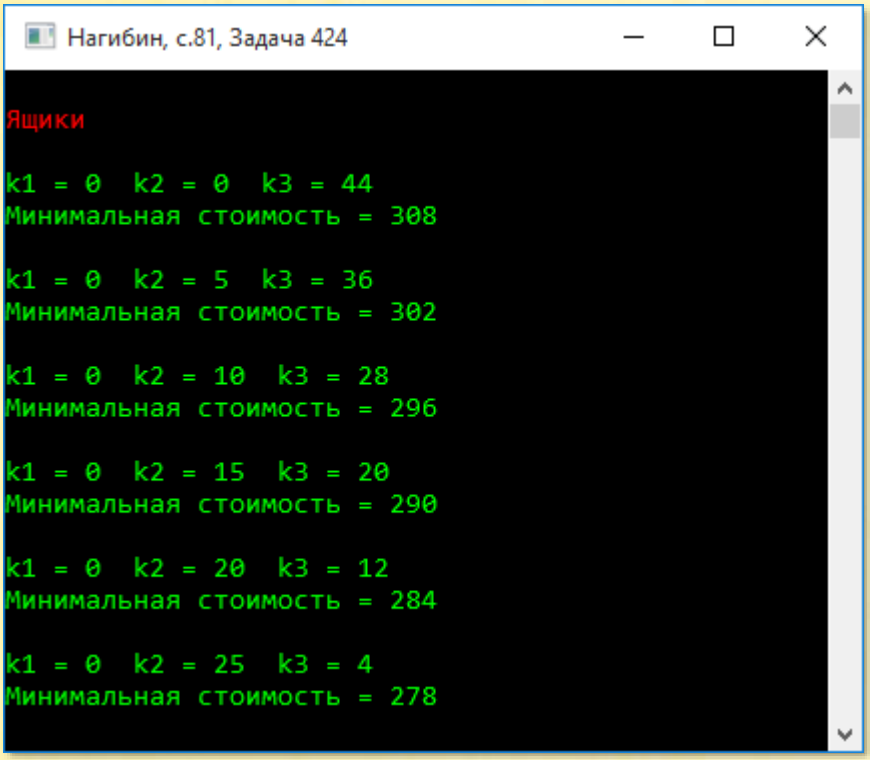
```
            // стоимость пересылки:
            var cost := k1 * 20 + k2 * 10 + k3 * 7;
            if (minCost >= cost) then
            begin
               minCost := cost;
               Console.WriteLine('k1 = {0}  k2 = {1}  k3 = {2}',
                                 k1, k2, k3);
               Console.WriteLine('Минимальная стоимость = {0}',
                                 minCost);
               Console.WriteLine();
            end
         end
      end
   end
end;
```

Если разные расфасовки дадут одинаковую стоимость, мы напечатаем все, чтобы узнать общее число решений задачи.

Поскольку результаты нашей упаковочной деятельности не ухудшаются по ходу работы программы, то минимальная стоимость окажется в самом низу списка, который вы видите на рисунке ниже.

Итак, **решение** задачи единственное:

Детали нужно упаковать в 25 ящиков второго типа и в 4 ящика третьего типа. При этом минимальная стоимость пересылки составит 278 рублей.



Путёвки

Процедура без параметров
Константы
Вложенные циклы for
Условный оператор if
Оператор continue
Форматированный вывод

Задача 425 из книги *Математическая шкатулка* [Нагибин88], страница 81:

Предполагается использовать 2000 р. на путёвки в дома отдыха.
Путёвки имеются на 15, 27 и 45 дней, стоимость их соответственно 21, 40 и 60 р.

Сколько и каких путёвок нужно купить, чтобы общее число дней отдыха было наибольшим?

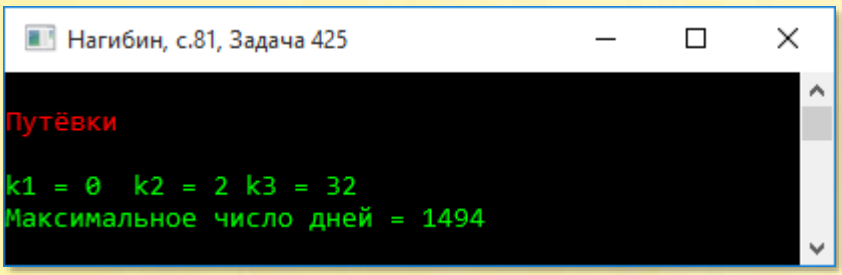
Эту задачу можно решить за пару минут, просто слегка изменив предыдущий проект:

```
uses
   System;

// Нагибин, с.81, Задача 424


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
const
   SUMMA = 2000;
   CK1 = SUMMA div 21;
   CK2 = SUMMA div 40;
   CK3 = SUMMA div 60;
begin
   // макс. число дней:
   var maxDays := 0;

   for var k1 := 0 to CK1 do
   begin
      for var k2 := 0 to CK2 do
      begin
         for var k3 := 0 to CK3 do
         begin
            if (k1 * 21 + k2 * 40 + k3 * 60 <> SUMMA)    then
               continue;

           // число дней отдыха:
           var days := k1 * 15 + k2 * 27 + k3 * 45;
           if (maxDays <= days) then
           begin
```

```
                maxDays := days;
                Console.WriteLine('k1 = {0}  k2 = {1} k3 = {2}',
                                  k1, k2, k3);
                Console.WriteLine('Максимальное число дней = {0}',
                                  maxDays);
                Console.WriteLine();
            end
        end
    end
end;


begin
    // заголовок окна:
    Console.Title := 'Нагибин, с.81, Задача 425';
    Console.WriteLine('');
    Console.ForegroundColor := ConsoleColor.Red;
    Console.WriteLine('Путёвки');
    Console.ForegroundColor := ConsoleColor.Green;
    Console.WriteLine();

    Solve();

    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```

Замечательно, что, решив одну задачу, потом можно использовать её код как заготовку при решении множества других подобных задач!

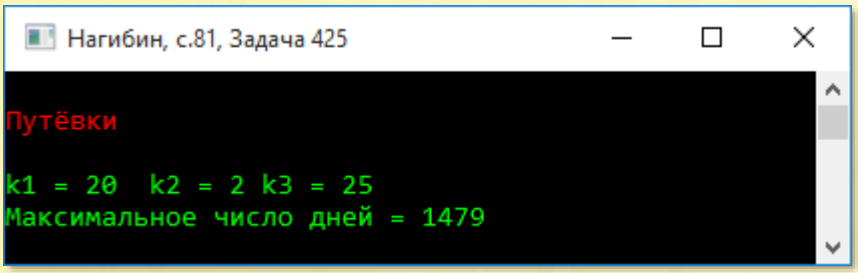
Наша программа советует не покупать путёвки на 15 дней, а сэкономленные деньги потратить на приобретение двух 27- и тридцати двух 45-дневных путёвок. Они позволят трудящимся заслуженно отдыхать 1494 дня:



В книге приведён другой ответ. Вероятно, автор не учёл, что можно вообще не покупать путёвки какого-либо вида. Чтобы проверить эту догадку, закомментированную строку замените другой:

```
//if (maxDays <= days) then
if ((maxDays <= days) and (k1*k2*k3 <> 0)) then
```

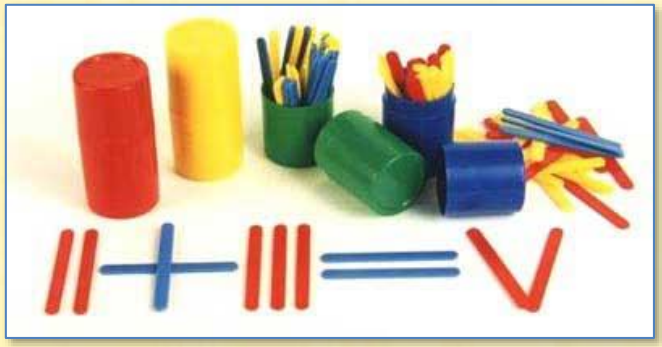
Теперь нулевые значения числа путёвок будут проигнорированы, и программа выдаст «книжные» результаты:



У автора задачи получилось на 15 дней меньше, чем у нас. Да, без компьютера толком не отдохнёшь!

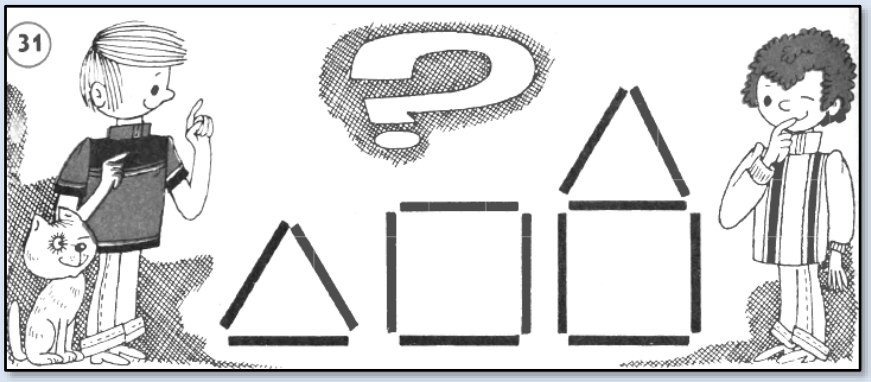
Счётные палочки

Процедуры без параметров
Константы
Вложенные циклы for
Условный оператор if
Оператор continue
Форматированный вывод



Задача 594 из книги *Математическая шкатулка* [Нагибин88], страница 97:

Из 36 счётных палочек построили треугольники, квадраты и домики – всего 10 фигур:



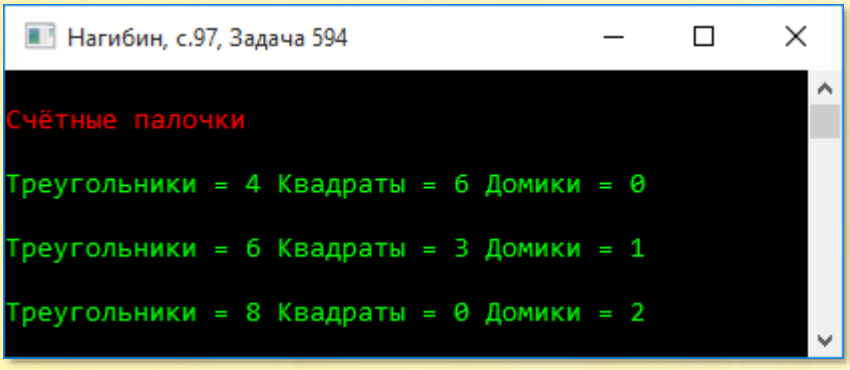
Найдите число фигур каждого вида.

```
uses
   System;

// Нагибин, с.97, Задача 594

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.97, Задача 594';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Счётные палочки');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Ясно, что из 36 палочек можно построить от 0 до 36/3 **треугольников**, от 0 до 36/4 **квадратов** и от 0 до 36/6 **домиков**. Все варианты строительства мы, как обычно, перебираем во вложенных циклах *for*:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
const
   PALOCHKI = 36;
   FIGUR = 10;
   CT = PALOCHKI div 3;
   CK = PALOCHKI div 4;
   CD = PALOCHKI div 6;

begin
   for var t := 0 to CT do
   begin
      for var k := 0 to CK do
      begin
         for var d := 0 to CD do
         begin
            if (t + k + d <> FIGUR) then
               continue;

            if (t * 3 + k * 4 + d * 6 <> PALOCHKI) then
               continue;

            Console.WriteLine('Треугольники = {0} Квадраты = {1}
                               Домики = {2}', t, k, d);
            Console.WriteLine();
         end
      end
   end
end;
```

Рисунок уверенно демонстрирует нам 3 решения задачи, хотя в книге указано только второе:



Мы опять можем предположить, что следовало построить **все** фигуры хотя бы по одному разу, хотя в книге это требование отсутствует.

Шестизначный перенос

Процедура без параметров
Цикл for
Константы
Оператор деления по модулю mod
Оператор деления div

Задача 557 из книги *Математическая шкатулка* [Нагибин88], страница 94:

Первая слева цифра шестизначного числа – 1. Если её перенести с первого места в конец числа, сохранив порядок остальных цифр, то вновь полученное число будет втрое больше первоначального.

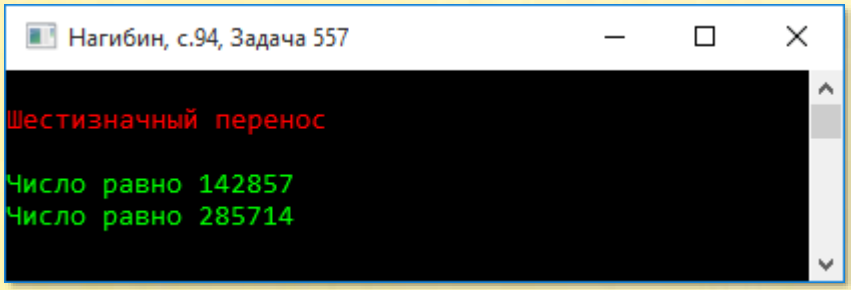
Восстановите первоначальное число.

В этой задача переносить нужно первую цифру:

```
uses
   System;

// Нагибин, с.94, Задача 557

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.94, Задача 557';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Шестизначный перенос');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
```

```
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
// мин. и макс. 6-значные числа:
const
   MIN6 = 100000;
   MAX6 = 999999;
begin
   for var num := MIN6 to MAX6 do
   begin
      // первая цифра числа:
      var start := num div 100000;
      // 5-значное число из последних 5 цифр числа num:
      var end5 := num mod 100000;
      // новое 6-значное число:
      var newnum := end5 * 10 + start;
      // это число должно быть втрое больше исходного:
      if (newnum <> 3 * num) then continue;
      Console.WriteLine('Число равно ' + num);
   end;
   Console.WriteLine();
end;
```

Среди 6-значных чисел, начинающихся с единицы, первоначальное число – единственное. Есть ещё одно подобное число, но оно начинается с двойки:



Факториальные нули

Процедура без параметров
Тип данных ***uint64***
Оператор ***break***
Бесконечный цикл ***while***

Задача 840 из книги *Математическая шкатулка* [Нагибин88], страница 128:

> Сколько нулей в конце записи числа, выражающего произведение
>
> 1 • 2 • 3 • 4 • 5 • 6 • . . . •14 • 15?

Произведение чисел 1..15 равно факториалу числа 15:

```
uses
   System;

// Нагибин, с.128, Задача 840

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.128, Задача 840';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Факториальные нули');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

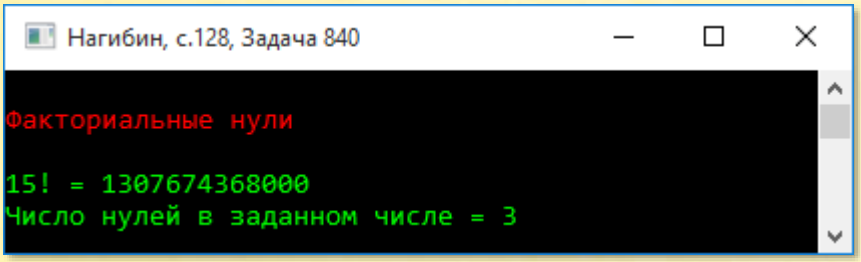
Достаточно пересчитать нули в конце записи числа 15!

В данном случае можно просто напечатать значение факториала на экране и посмотреть, сколькими нулями заканчивается запись. Но вполне вероятно, что вам встретится и другая задача на подсчёт хвостовых нулей, поэтому мы добросовестно пересчитаем их в процедуре Solve:

```
// ВЫЧИСЛЯЕМ ФАКТОРИАЛ ЗАДАННОГО ЧИСЛА
function factorial(num: integer): uint64;
begin
   if (num = 0) then
   begin
      Result := 1;
      exit;
   end;

   var fact: uint64 := 1;
   for var i: uint64 := 2 to uint64(num) do
      fact *= i;
   Result := fact;
end;

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var num := 15;
   // находим факториал заданного числа:
   var fact := factorial(num);
   // печатаем факториал:
   Console.WriteLine(num + '! = ' + fact.ToString());
   // считаем нули:
   var n := 0;
   while (true) do
   begin
      // очередная цифра сзади:
      var dig := fact mod 10;
      if (dig <> 0) then
         break;
      fact := fact div 10;
      n += 1;
   end;
   Console.WriteLine('Число нулей в заданном числе = ' + n);
end;
```

Последнюю цифру легко отделить от числа с помощью оператора **%**. Чтобы предпоследняя цифра числа обратилась в последнюю, нужно разделить его на 10. Как только последняя цифра будет отличаться от нуля, подсчёт нужно закончить.

На рисунке хорошо видно, что факториал числа 15 заканчивается тремя нулями:



Гаусс

Процедуры с параметрами
Тип данных ***int64***
Цикл ***for***

Задача 5 из книги *Математическая шкатулка* **[Нагибин88]**, страница 15:

> Рассказывают, что в начальной школе, где учился мальчик Карл Гаусс, ставший потом знаменитым математиком, учитель, чтобы занять класс на продолжительное время самостоятельной работой, дал детям такое задание - вычислить сумму всех натуральных чисел от 1 до 100. Но маленький Гаусс это задание моментально выполнил.
>
> **Попробуй и ты быстро выполнить это задание.**

Нетрудно в этом ряде чисел увидеть **арифметическую прогрессию**, начинающуюся с единицы и насчитывающую 100 членов. Разность арифметической прогрессии равна 1. Зная все параметры этого ряда, мы быстро найдём его сумму:

Сумма = (1 + 100)*100/2 = 5050

```
uses
   System;

// Нагибин, с.15, Задача 5

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.15, Задача 5';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Гаусс');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve(1,100,1);

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Учитель, конечно, предполагал, что школяры будут вычислять сумму ряда последовательным прибавлением очередного члена ряда к текущей сумме.

Давайте в процедуре Solve потакнём недалёкому учителю, тем более что компьютер – не Гаусс и быстро, а главное – с удовольствием найдёт сумму именно так.

Пусть сумма ряда вначале равна нулю:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve(start, num, r: integer);
begin
   // сумма ряда:
   var sum: int64 := 0;
```

Текущий член ряда – это сначала первый член прогрессии, который мы назовём start:

```
// текущий член ряда:
var n := start;
```

Если в последовательности num членов, то мы их друг за другом добавляем к сумме:

```
for var i := 0 to num - 1 do
begin
   sum += n;
```

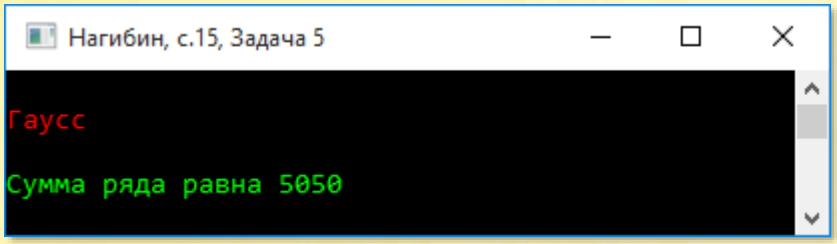
При этом не забываем вычислять следующий член последовательности, прибавляя к текущему разность арифметической прогрессии:

```
     n += r;
   end;

   Console.WriteLine('Сумма ряда равна {0}', sum);

   Console.WriteLine();
end;
```

В нашей задаче:

- start = 1
- num = 100
- r = 1

Наша глуповатая, но прилежная процедура правильно подсчитала сумму заданного ряда:



Зато наш метод получился универсальным! Подставляя нужные значения в процедуру *Solve*, вы быстро найдёте сумму любой арифметической прогрессии. Например, давайте решим такую задачу.

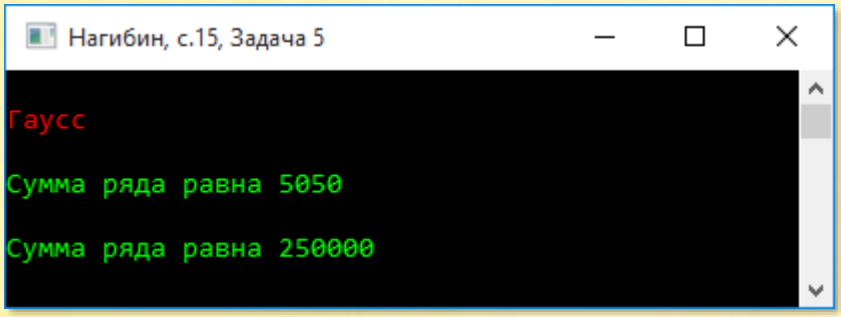
Задача 7.1 из книги *Математическая шкатулка* **[Нагибин88]**, страница 15:

> Как быстро вычислить сумму 1 + 3+ 5+7 + 9 + … + 997 +999?

Добавьте в **главный блок** выделенную строку:

```
Solve(1, 100, 1);
Solve(1,999 div 2+1,2);
```

И получите **ответ** на задачу:



Здесь главное – не ошибиться со значениями параметров в процедуре *Solve*!

- start = 1
- num = 999/2+1

- r = 2

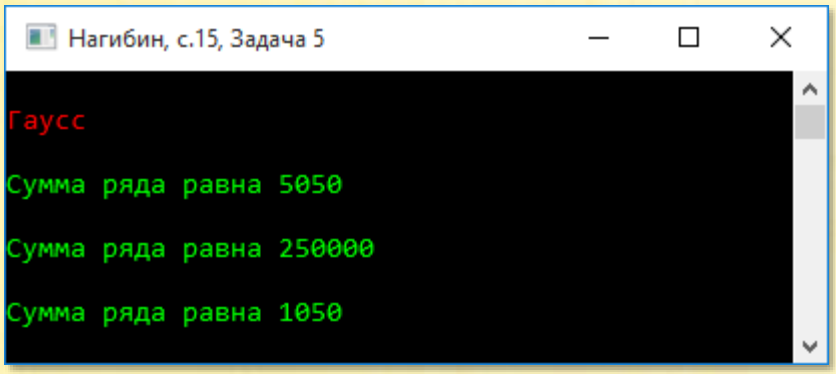
При вычислении num следует добавить 1, так как при целочисленном делении результат будет округлён отбрасыванием целой части, поэтому число членов окажется на 1 меньше действительного.

Задача 838 из книги *Математическая шкатулка* [Нагибин88], страница 128:

Вычислите: 5 + 10+ 15+20 + 25 + … + 100.

И опять одна строка программы решает задачу:

```
Solve(1, 100, 1);
Solve(1,999 div 2+1,2);
Solve(5,100 div 5,5);
```



В этой задаче:

- start = 5
- num = 100/5
- r = 5

Плюс-минус

Процедура без параметров
Тип данных int64
Цикл for
Комбинированные операторы присваивания

Задача 7-2 из книги *Математическая шкатулка* [Нагибин88], страница 15:

> Как быстро вычислить:
>
> 99 - 97 + 95 - 93 + 91 - 89 + ... + 7 - 5 + 3 – 1?

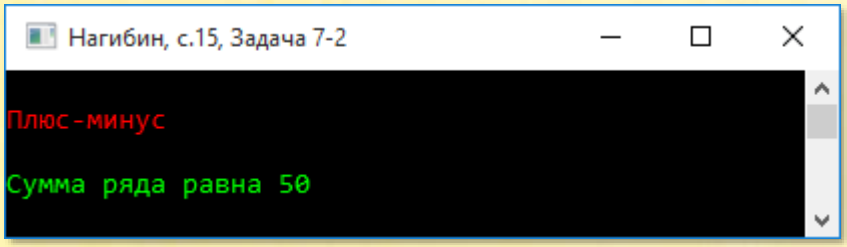
Конечно, быстро вычислить можно на компьютере!

```
uses
   System;

// Нагибин, с.15, Задача 7-2

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.15, Задача 7-2';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Плюс-минус');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

По сравнению с обычной арифметической прогрессией (здесь она убывающая) знак членов ряда чередуется: плюс-минус-плюс-минус...

Но достаточно ввести переменную sign, чтобы решить и эту задачу:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   // сумма ряда:
   var sum: int64 := 0;
   var start := 99;
   // текущий член ряда:
   var n := start;
   var r := -2;
   var num := 99 div 2+1;
   // знак алгебраической суммы:
   var sign := 1;

     for var i := 0 to num-1 do
     begin
         sum := sum + n*sign;
         n += r;
         // меняем знак:
         sign *= -1;
   end;

   Console.WriteLine('Сумма ряда равна {0}', sum);
   Console.WriteLine();
end;
```



К сожалению, в издании 1988 года отсутствует и ответ, и решение этой задачи, а вот в более раннем издании 1958 года приведено решение – и весьма остроумное!

Минус-плюс

Процедура без параметров
Тип данных ***int64***
Цикл ***for***

Задача 482-8 из книги *Математическая шкатулка* [Нагибин88], страница 88:

Найдите простой приём вычисления:

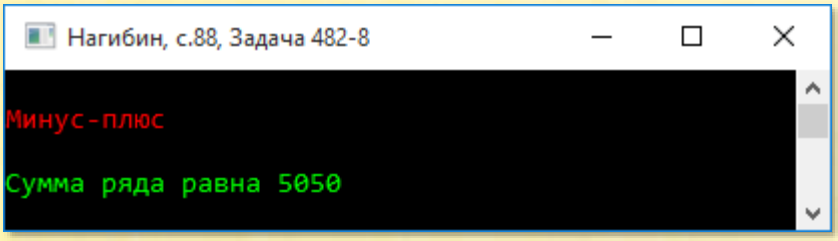
$100^2 - 99^2 + 98^2 - 97^2 + 96^2 - 95^2 + \ldots + 4^2 - 3^2 + 2^2 - 1^2$

Здесь мы видим практически тот же ряд, что и в предыдущей задаче, только ещё проще!

```
uses
   System;

// Нагибин, с.88, Задача 482-8

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.88, Задача 482-8';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Минус-плюс');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Задача решается без труда, но довольно интересно, что ответ на неё полностью совпадает с тем ответом, который мы получили, решая задачу Гаусса:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   // сумма ряда:
   var sum := 0;
   // знак алгебраической суммы:
   var sign := 1;

   for var i := 100 downto 1 do
   begin
        sum += i*i*sign;
        // меняем знак:
        sign *= -1;
   end;
   Console.WriteLine('Сумма ряда равна {0}', sum);
   Console.WriteLine();
end;
```



Дробный ряд

Процедура без параметров
Тип данных ***double***
Цикл ***for***
Комбинированные операторы присваивания
Оператор деления ***/***

Задача 9-1 из книги *Математическая шкатулка* **[Нагибин88]**, страница 16:

Найдите простой приём вычислений и воспользуйтесь им для вычисления суммы:

$$\frac{1}{1\cdot2}+\frac{1}{2\cdot3}+\frac{1}{3\cdot4}+\frac{1}{4\cdot5}+\frac{1}{5\cdot6}+\frac{1}{6\cdot7}+\frac{1}{7\cdot8}+\frac{1}{8\cdot9}+\frac{1}{9\cdot10}$$

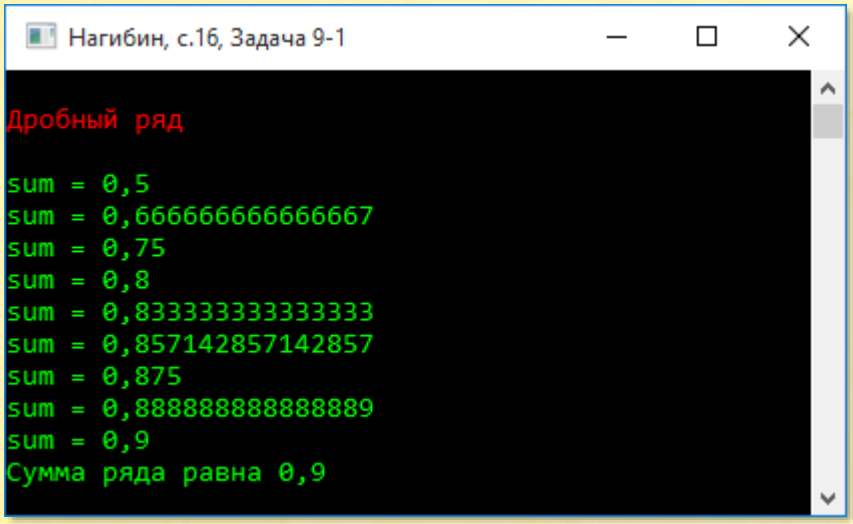
```
uses
   System;

// Нагибин, с.16, Задача 9-1

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.16, Задача 9-1';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Дробный ряд');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Этот ряд – почти арифметическая прогрессия: в знаменателе первый сомножитель изменяется от 1 до 9 с приращением 1. Второй сомножитель равен первому, плюс 1. Произведение этих сомножителей, а значит и дробей, мы легко найдём. Осталось вычислить сумму дробей:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
```

```
    // сумма ряда:
    var sum:= 0.0;
    // текущий член ряда:
    var n := 1;
    var r := 1;
    var num := 9;

     for var i := 0 to num-1 do
     begin
         sum += 1.0/(n*(n+1));
         n += r;
         Console.WriteLine('sum = {0}', sum);
    end;

    Console.WriteLine('Сумма ряда равна {0}', sum);
    Console.WriteLine();
end;
```

Рисунок показывает, что сумма равна 0,9, и подсказывает нам, что у этой задачи имеется и красивое некомпьютерное решение. Найдите его!



Ещё один дробный ряд

Процедура без параметров
Тип данных ***double***
Цикл ***for***
Комбинированные операторы присваивания
Оператор деления ***/***

Задача 9-2 из книги *Математическая шкатулка* [Нагибин88], страница 16:

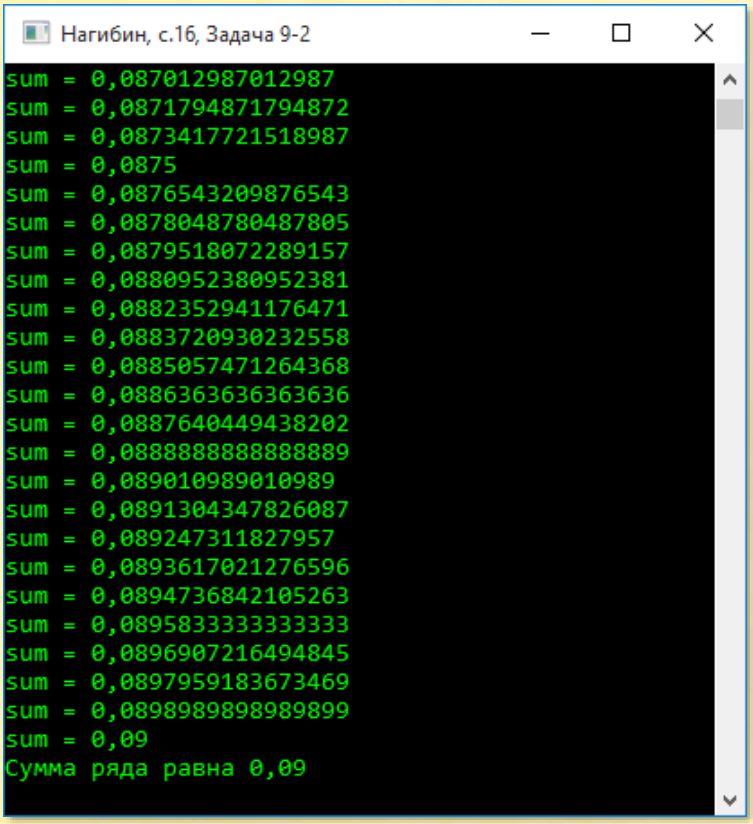
Найдите простой приём вычислений и воспользуйтесь им для вычисления суммы:

$$\frac{1}{10\cdot 11}+\frac{1}{11\cdot 12}+\frac{1}{12\cdot 13}+\frac{1}{13\cdot 14}+\frac{1}{14\cdot 15}+\ldots+\frac{1}{98\cdot 99}+\frac{1}{99\cdot 100}$$

```
uses
   System;

// Нагибин, с.16, Задача 9-2

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.16, Задача 9-2';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Ещё один дробный ряд');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Задача очень похожа на предыдущую, поэтому решается исправлением двух строк в коде:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   //сумма ряда:
   var sum:= 0.0;
   //текущий член ряда:
   var n := 10;
   var r := 1;
   var num := 90;
   for var i := 0 to num-1 do
   begin
        sum += 1.0/(n*(n+1));
        n += r;
        Console.WriteLine('sum = {0}', sum);
   end;
   Console.WriteLine('Сумма ряда равна {0}', sum);
   Console.WriteLine();
end;
```



Трёхзначное число 2

Процедура с параметрами
Цикл ***for***
Вложенные операторы ***if***
Оператор деления ***div***
Оператор целочисленного деления ***mod***
Оператор ***break***

Задача 597 из книги *Математическая шкатулка* [Нагибин88], страницы 97-98:

Сколько слагаемых суммы 1 + 2 + 3 + 4 + 5 + … надо взять, чтобы получить трёхзначное число, состоящее из одинаковых цифр?

```
uses
   System;

// Нагибин, с.97-98, Задача 597

begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.97-98, Задача 597';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Трёхзначное число');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve(1,100,1);

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

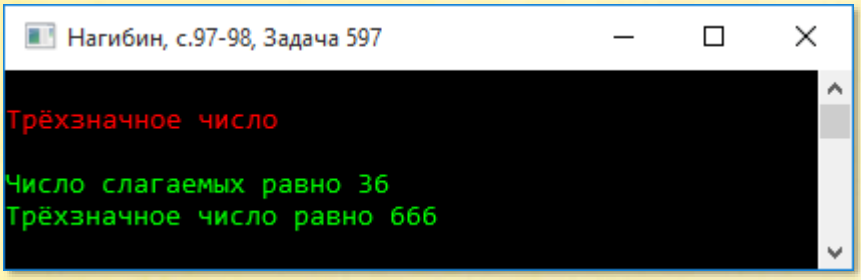
Поскольку числовой ряд представляет собой арифметическую прогрессию, сумму которой мы умеем находить, то нам нужно только проследить, когда эта сумма превратится в трёхзначное число с одинаковыми цифрами.

Искать заданное число можно в бесконечном цикле *for*, который следует прервать по исчерпании всех трёхзначных сумм:

```
procedure Solve(start, num, r: integer);
begin
  // сумма ряда:
  var sum := 0;
  // текущий член ряда:
  var n := start;
  for var i := 1 to integer.MaxValue do
  begin
        sum += n;
        if (sum >= 111) then
        begin
              // число единиц:
              var e := sum mod 10;
              // число десятков:
              var d := sum div 10 mod 10;
              // число сотен:
              var s := sum div 100;
              if ((e = d) and (e = s)) then
              begin
                    Console.WriteLine('Число слагаемых равно {0}', i);
                    Console.WriteLine('Трёхзначное число равно {0}',
                                sum);
              end
        end;
        // трёхзначные числа закончились:
        if (sum > 999) then break;
        n += r;
  end;
  Console.WriteLine();
end;
```

Правда, по условиям задачи, цикл можно закончить сразу как только будет найдено первое трёхзначное число с одинаковыми цифрами, но мы проверим все

трёхзначные суммы – в надежде, что среди них найдутся и другие подходящие. Ан нет: рисунок разочаровывает нас – задача имеет *единственное* решение!



Сотая цифра

Процедура без параметров
Цикл **while**
Вложенные циклы **while**
Условный оператор **if**
Оператор **break**
Массив **integer[]**
Комбинированные операторы присваивания
Оператор цикла **foreach**

Задача 300 из книги *Математическая шкатулка* [Нагибин88], страница 45:

> Все натуральные числа, начиная с 1, записаны в порядке их возрастания: 1 2 3 4 5 6 7 8 9 10 11…
>
> Какая цифра в этой записи стоит на сотом месте?

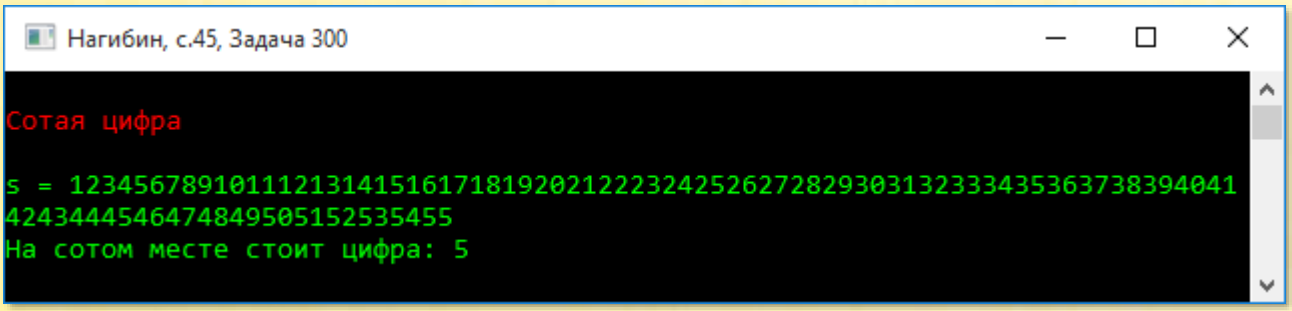
Проще всего решить задачу «строковым» способом:

```
uses
   System;

// Нагибин, с.45, Задача 300
```

```
begin
   // заголовок окна:
   Console.Title := 'Нагибин, с.45, Задача 300';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Сотая цифра');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();
   Solve();
   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var s := String.Empty;
   var i := 1;
   while(s.Length < 100) do
   begin
       s += i;
       i += 1;
   end;
   Console.WriteLine('s = {0}', s);
   Console.WriteLine('На сотом месте стоит цифра: {0}', s[100]);
   Console.WriteLine();
end;
```

Пока в строке меньше 100 знаков, мы добавляем к ней следующее число. Поскольку при этом длина строки может оказаться больше 100 символов, то мы просто интересуемся, какая цифра стоит на **сотом** месте (её индекс - 100).

На рисунке вы видите, что сотое место в числовой записи занимает цифра 5:



Но строковый способ решения задачи не совсем честный, поэтому давайте обойдёмся без строк - только числами!

Процедура Solve при этом, конечно, станет длиннее и сложнее, но зато всё по-честному!

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve2();
begin
   var num := 1;
   var nd := 0;
   var anum := new integer[101];
   while (nd <= 101) do
   begin
        var n := num;
        var len := NumDigit(num);
        while (n > 0) do
        begin
             if (nd >= 101) then  break;
             // очередная цифра:
             anum[nd + len-1] := n mod 10;
             len -= 2;
             n := n div 10;
             nd += 1;
        end;
        num += 1;
        if (nd >= 101) then  break;
   end;
   var s := String.Empty;
   foreach var i in anum do
        s += i;
   Console.WriteLine('s = {0}', s);
   Console.WriteLine('На сотом месте стоит цифра: {0}', anum[99]);
   Console.WriteLine();
end;
```

Здесь число num увеличивается от 1 и до тех пор, пока число цифр в массиве *anum* не достигнет сотни.

Теперь нам нужно самостоятельно выделять из каждого числа его цифры и заносить в массив anum. После этого мы формируем из элементов массива строку, но исключительно для того, чтобы напечатать её на экране; для решения задачи она не нужна.

Ответ мы получим, естественно, тот же самый, что и раньше.

Задания для самостоятельного решения

Задача #24

Математическая шкатулка

Какое целое число делится (без остатка) на любое целое число, отличное от 0?.

Ответ: 0

Задача #25

Математическая шкатулка

Сумма каких двух натуральных чисел равна их произведению?

Ответ: 2 + 2 = 2 x 2.

Задача #33

Математическая шкатулка

Какую последнюю цифру может иметь квадрат натурального числа? Куб его? Четвёртая степень?

Ответ.

Квадраты *могут оканчиваться на 1, 4, 5, 6, 9.*
Кубы *могут оканчиваться на 1, 2, 3, 4, 5, 6, 7, 8, 9.*
Четвёртые степени *могут оканчиваться на 1, 5, 6.*

Задача #34

Математическая шкатулка

Могут ли числа 458, 523, 652 быть квадратами или кубами целого числа?

Ответ: Нет.

Задача #297

Математическая шкатулка

Чтобы пронумеровать страницы некоторой книги, понадобилось 1164 цифры.

Сколько в этой книге страниц?

Ответ: 424

Задача #298

Математическая шкатулка

Сколько цифр нужно употребить для нумерации книги, в которой 634 страницы?

Ответ: 1794

Задача #24

Математическая шкатулка

Какое целое число делится (без остатка) на любое целое число, отличное от 0?.

Ответ: 0

Задача #25

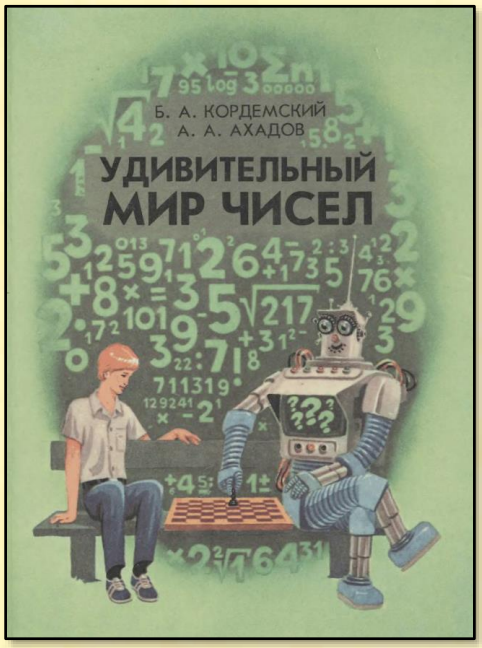
Математическая шкатулка

Сумма каких двух натуральных чисел равна их произведению?

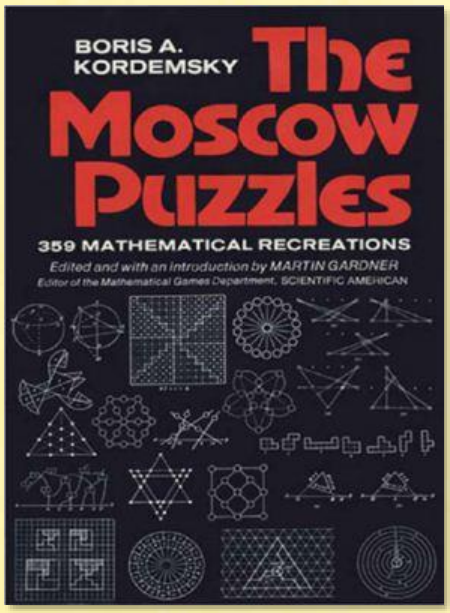
Ответ: 2 + 2 = 2 x 2.

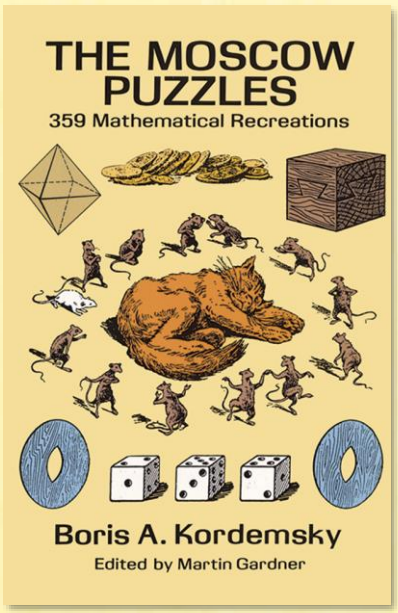
Задачи Бориса Кордемского

Ещё одна очень известная книга по занимательной математике - ***Удивительный мир чисел*** – принадлежит перу Бориса Анастасьевича Кордемского (в соавторстве с Аскером Ахадовым)



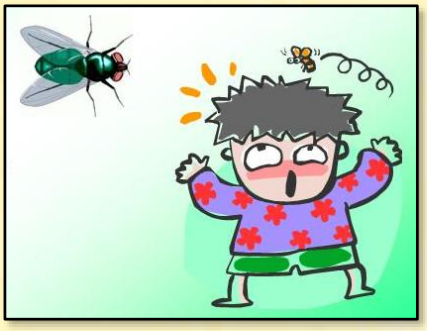
Избранные задачи Бориса Кордемского были изданы на английском языке под названием: *Boris A. Kordemsky. The Moscow Puzzles: 359 Mathematical Recreations*. Редактировал книгу Мартин Гарднер.





Назойливый остаток

Вложенные циклы ***for***
Бесконечный цикл ***for***
Условный оператор ***if***
Оператор ***mod***
Оператор ***break***
Оператор ***continue***



В книге Бориса Кордемского и Аскера Ахадова *Удивительный мир чисел* [КА86] вы найдёте немало интересных задач, в том числе и на делимость. На странице 86 авторы предлагают решить задачу **Назойливый остаток**:

> Некоторые числа, кратные числу 7, при делении на 2, на 3, на 4, на 5 и на 6 дают остаток 1.
>
> **Найдите наименьшее из таких чисел.**

Эта же задача напечатана в книге Фёдора Нагибина и Евгения Канина *Математическая шкатулка* [Нагибин88], задача 42, страницы 18-19, но в более занимательной форме:

> Колхозница привезла на рынок для продажи корзину яиц. Продавала она их по одной и той же цене. После продажи яиц колхозница пожелала проверить, верно ли она получала деньги. Но вот беда: она забыла, сколько у неё было яиц. Вспомнила она только, что когда перекладывала яйца по 2, то оставалось одно яйцо; одно яйцо оставалось также при перекладывании яиц по 3, по 4, по 5, по 6. Когда же она перекладывала яйца по 7, то не оставалось ни одного.
>
> **Помоги колхознице сообразить, сколько у неё было яиц.**

Бросаемся на помощь колхознице и в **главном блоке** программы вызываем процедуру *Solve* для решения этой головоломки:

```
uses
   System;

// Кордемский, с.86, Задача 8

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.86, Задача 8';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Назойливый остаток');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Поскольку речь идёт о **натуральных** числах, то мы можем начать наши поиски с *единицы*:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
// мин. число:
const MIN = 1;
begin
```

Если число 1 не является решением задачи, то мы переходим к двойке, затем к тройке, и так далее – пока не найдётся искомое число **num**.

Так как мы всякий раз добавляем к начальному значению переменной **min** единицу, то вполне разумно делать это в цикле ***for***, в котором переменная **num** и будет играть роль *переменной цикла*.

Обычно к циклу ***for*** прибегают тогда, когда число повторов точно известно. В нашем же случае, мы знаем только начало цикла (*min = 1*), но не знаем, на каком

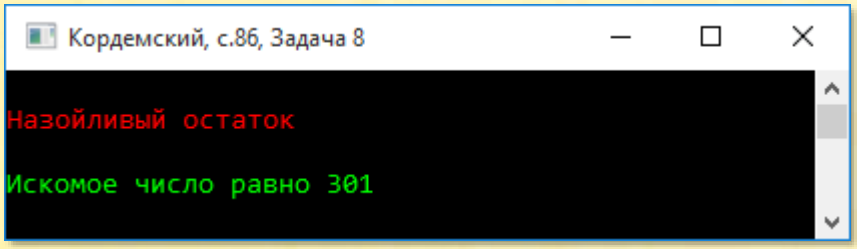
числе он закончится. Как обычно, мы организуем **бесконечный цикл**, который прервём сразу же, как только искомое число будет обнаружено. Для этого мы воспользуемся *флажком* – логической переменной **flg**. Если очередное число *num* выдержит все проверки, значение флага останется верным (*true*), а мы с помощью оператора *break* прервём цикл *for* и напечатаем решение задачи в Консольном окне:

```
var num := 0;
 var flg := true;

 begin
    flg := true;
    // число не кратно семи:
    if (num mod 7 <> 0) then continue;
```

Проверку очередного числа на остаток, равный единице, также можно проводить в цикле, чтобы не выписывать несколько одинаковых условий. Если число *num* при делении хотя бы на одно из чисел 2..6 не даёт в остатке 1, то мы сбрасываем флажок и прерываем **внутренний** цикл *for*:

```
    for var d := 2 to 6 do
    begin
       if (num mod d <> 1) then
       begin
          flg := false;
          break;
       end
    end;
    if (flg) then break;
 end;
 Console.WriteLine('Искомое число равно ' + num);

 Console.WriteLine();
end;
```

На рисунке вы видите **ответ** на эту задачу:



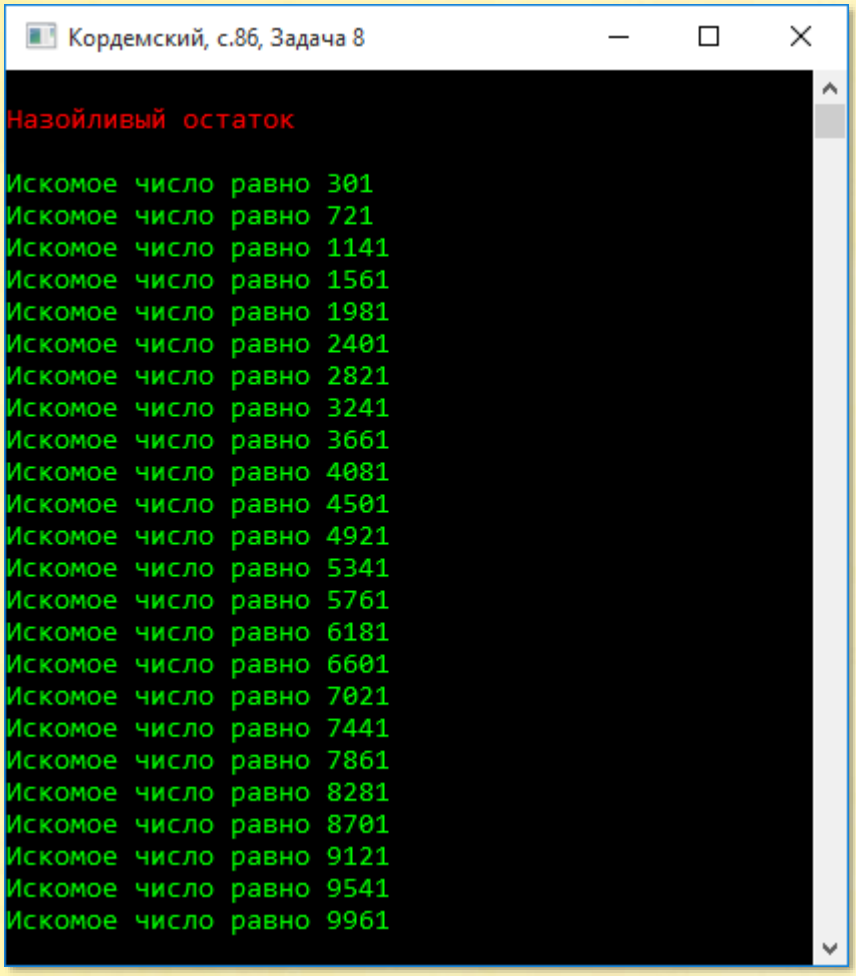
Не знаю, как вам, но мне интересно, а *есть ли ещё и другие числа, которые имеют назойливый остаток*?

Имея компьютер, мы легко утолим своё любопытство – достаточно написать новую процедуру Solve2:

```
procedure Solve2();
const
   // мин. число:
   MIN = 1;
   // макс. число:
   MAX = 10000;
begin
   var num := 0;
   var flg := true;
   for num := MIN to MAX do
   begin
      flg := true;
      // число не кратно семи:
      if (num mod 7 <> 0) then continue;

      for var d := 2 to 6 do
      begin
         if (num mod d <> 1) then
         begin
            flg := false;
            break;
         end
      end;
      if (flg) then
         Console.WriteLine('Искомое число равно ' + num);
   end;
   Console.WriteLine();
end;
```

Тут, конечно, следует учесть, что бесконечный цикл уже не годится, потому что он действительно станет бесконечным, если искомых чисел очень много (а это нетрудно предвидеть).

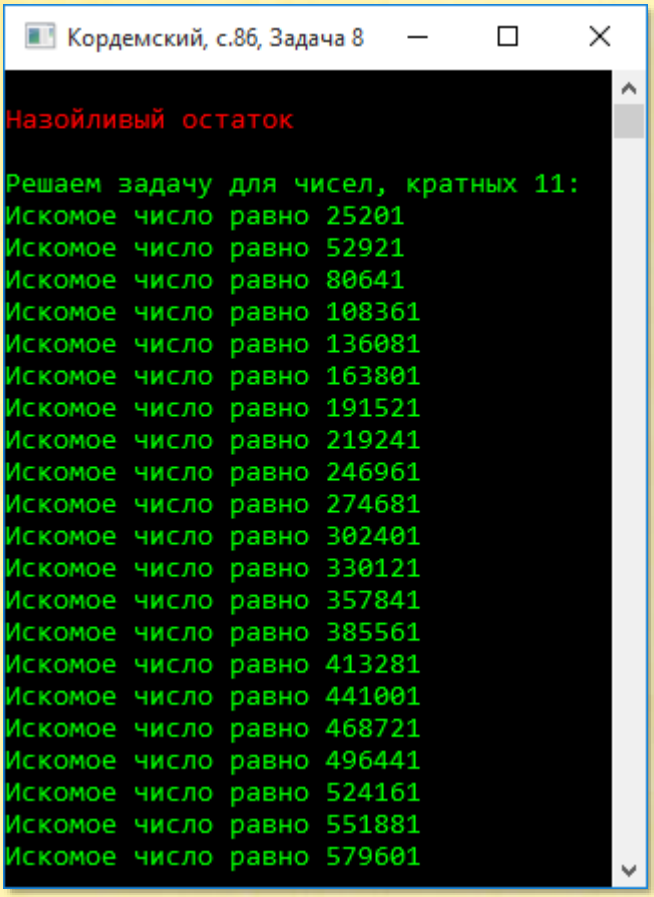
Запускаем приложение и видим, что среди первой десятки тысяч натуральных чисел довольно много подходящих под условие задачи:

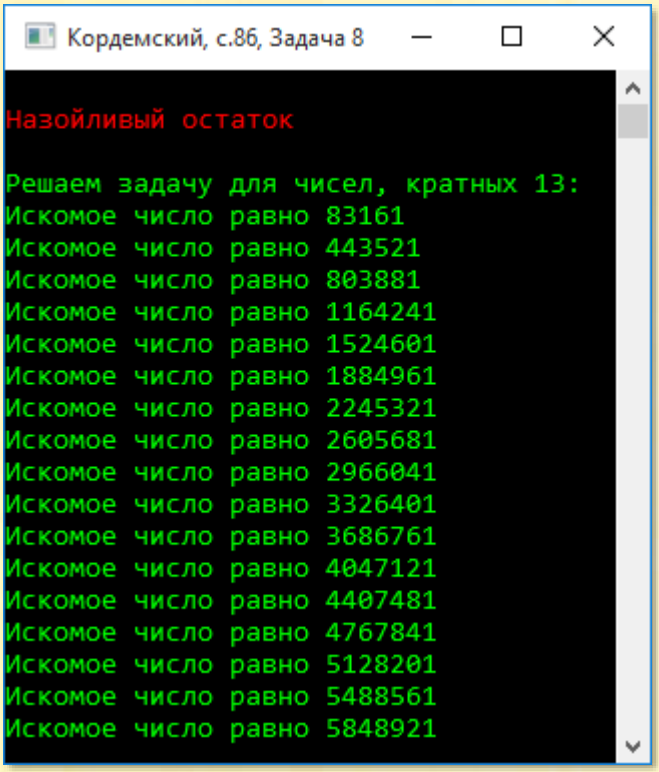


Также нетрудно подметить такую **закономерность**. Если обозначить через **n** номер искомого числа, то все числа с назойливыми остатками можно легко найти по формуле:

$$num = 301 + 420(n-1)$$

Любопытно, что, если решать задачу для чисел, кратных не 7, а б*о*льшим числам, то это непременно должны быть **простые** числа. Например, решения для чисел 11 и 13 показаны на рисунке:





Во Франции бытует подобная задач (*Математический фольклор, Задача Д17*):

> Женщина несла на базар две корзины яиц. Прохожий случайно толкнул её, корзинки упали и яйца разбились. Виновник извинился, предложил возместить ущерб и спросил:
>
> - Сколько яиц было в корзинах?
> - Не помню точно, - ответила женщина, - но знаю, когда вынимала их по 2, по 3, по 4, по 5 или по 6 яиц, то в корзине оставалось по 1 яйцу, а когда вынимала по 7, то не оставалось ни одного.
>
> Сколько яиц было в корзинах?

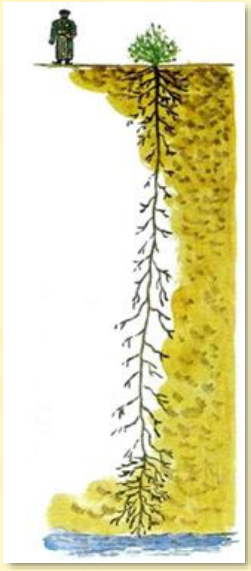
Популярна яичная тема и в Болгарии (*Математический фольклор, Задача 87*):

Женщина несла на базар две корзины яиц. Её нечаянно толкнул парень, корзины упали, а яйца разбились. Парень, чтобы заплатить, спросил, сколько было всего яиц.

- Я их не считала, но когда складывала в корзины по 2, по 3, по 4, по 5, по 6, то всякий раз оставалось по 1 яйцу, а когда складывала по 7 - не оставалось ни одного.

Сколько яиц было в корзинах?

Длинные корни



Задача 1 из книги *Удивительный мир чисел* [КА86], страница 98:

Путём преобразования выражений

а) $\sqrt{1982 \cdot 1983 \cdot 1984 \cdot 1985 + 1}$,

б) $\sqrt{1979 \cdot 1986 \cdot 1993 \cdot 2000 + 2401}$

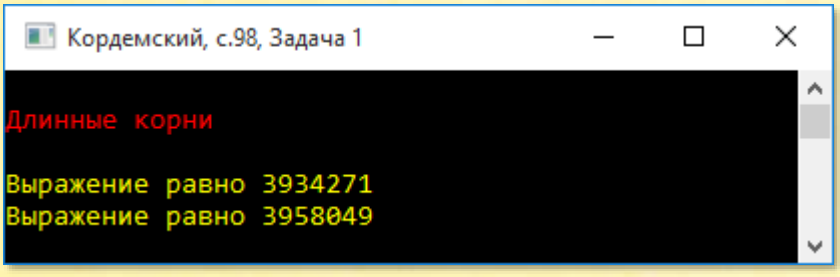
найдите их точные значения.

Основная сложность в решении этой задачи – вычислить произведение четырёх чисел. Сами по себе все сомножители имеют тип *inegert*, а их произведение должно иметь тип int64, так как оно слишком велико для типа *integer*. Поэтому нужно привести первый числовой литерал к длинному типу:

```
uses
   System;

// Кордемский, с.98, Задача 1
```

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
    Console.ForegroundColor := ConsoleColor.Yellow;

    var p := int64(1982) * 1983 * 1984 * 1985;
    var res := System.Math.Sqrt(p + 1);
    Console.WriteLine('Выражение равно {0}', res);

    p := int64(1979) * 1986 * 1993 * 2000;
    res := System.Math.Sqrt(p + 2401);
    Console.WriteLine('Выражение равно {0}', res);

    Console.WriteLine();
end;


begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.98, Задача 1';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Длинные корни ');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

В этом случае произведение получит тип *int64*, и вычисления пройдут без ошибок:



Каковы жуки?

Функция без параметров
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор ***exit***
Оператор ***or***

Задача 9 из книги *Удивительный мир чисел* [КА86], страница 100:

> **Решите уравнение** в целых неотрицательных числах:
>
> 520•(Ж•У•К•И + Ж•У + Ж•И + К•И + 1) = 577•(У•К•И + У + И).

Числовые ребусы вполне успешно можно решать **методом полного перебора**. В них каждая цифра заменена буквой, но, поскольку цифр только десять, то каждая буква может принимать всего 10 разных значений - от 0 до 9. В любом ребусе не более десятка разных букв, то есть в худшем случае нам придётся проверить 10 000 000 000 вариантов. Современным компьютерам это вполне по силам. Впрочем, так бездумно компьютер не используют, поэтому даже при полном переборе следует разумно ограничивать число вариантов. Обычно сделать это очень просто, поскольку некоторые способы решения криптарифмов лежат на поверхности и не требуют глубоких размышлений. Например, для уменьшения числа вариантов достаточно учесть тот очевидный факт, что все буквы должны иметь **разное** значение. Это следует из условия самой головоломки: *разным буквам соответствуют разные цифры*. Вот теперь можно смело браться за любой числовой ребус.

Главный блок просто вызывает функцию *Solve*, а затем печатает число найденных решений:

```
uses
```

```
   System;

// Кордемский, с.100, Задача 9

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.100, Задача 9';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Каковы жуки?');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                        nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

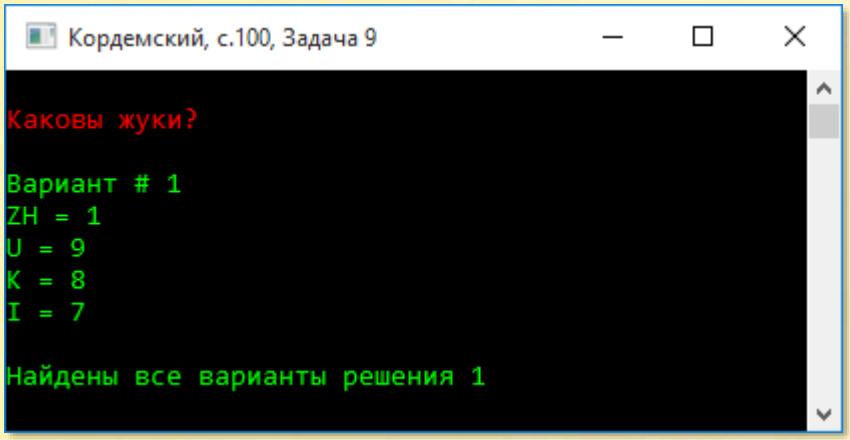
Функция Solve, как это обычно и бывает при полном переборе, очень простая. Записываем столько вложенных циклов *for*, сколько разных букв в ребусе. В нашем примере всего 4 разные буквы, поэтому и циклов тоже будет 4. Начиная со второй буквы, делаем проверку: её цифровое представление не должно совпадать с предыдущими буквами. Найдя значение каждой буквы, проверяем условие:

```
520 * (ZH * U * K * I + ZH * U + ZH * I + K * I + 1) =
                                        577 * (U * K * I + U + I)
```

Если оно выполняется (то есть левая часть выражения равна правой), то мы выписываем найденное решение в консольном окне, а затем ищем другие решения:

```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   //520*(ZH*U*K*I + ZH*U + ZH*I + K*I + 1) = 577*(U*K*I + U + I)
   Result := 0;
   for var ZH := 0 to 9 do
```

```
        for var U := 0 to 9 do
        begin
           if (U = ZH) then continue;
           for var K := 0 to 9 do
           begin
              if ((K = U) or (K = ZH)) then continue;
              for var I := 0 to 9 do
              begin
                 if ((I = K) or (I = U) or (I = ZH)) then continue;
                 if (520 * (ZH * U * K * I + ZH * U + ZH * I + K * I
                           + 1) = 577 * (U * K * I + U + I)) then
                 begin
                    Result += 1;
                    Console.WriteLine('Вариант # ' +
                                      result.ToString());
                    var s := 'ZH = ' + ZH;
                    Console.WriteLine(s);
                    s := 'U = ' + U;
                    Console.WriteLine(s);
                    s := 'K = ' + K;
                    Console.WriteLine(s);
                    s := 'I = ' + I;
                    Console.WriteLine(s);

                    Console.WriteLine();
                 end
              end
           end
        end
end;
```



В итоге мы найдём *единственное* решение этого ребуса:

Точно так же вы можете решить любой числобус, НО – каждый раз вам придётся писать **новую** функцию *Solve*! Такова плата за простоту программы…

Четыре "пари"

Функция без параметров
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор ***exit***
Оператор ***or***

Задача 8 из книги *Удивительный мир чисел* [КА86], страница 100:

Как велико ПАРИ, если:

$$\begin{cases} \dfrac{\text{П}\cdot\text{А}\cdot\text{Р}\cdot\text{И}}{\text{П}+\text{А}+\text{Р}} = 5, & \dfrac{\text{П}\cdot\text{А}\cdot\text{Р}\cdot\text{И}}{\text{И}+\text{Р}+\text{А}} = 3, \\ \dfrac{\text{П}\cdot\text{А}\cdot\text{Р}\cdot\text{И}}{\text{П}+\text{И}+\text{Р}} = \dfrac{10}{3}, & \dfrac{\text{П}\cdot\text{А}\cdot\text{Р}\cdot\text{И}}{\text{П}+\text{А}+\text{И}} = \dfrac{15}{4}. \end{cases}$$

В этой системе числовых ребусов также необходимо найти цифровые значения четырёх букв, поэтому бо́льшую часть кода мы можем скопировать из предыдущего проекта.

```
uses
   System;

// Кордемский, с.100, Задача 8

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.100, Задача 8';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
```

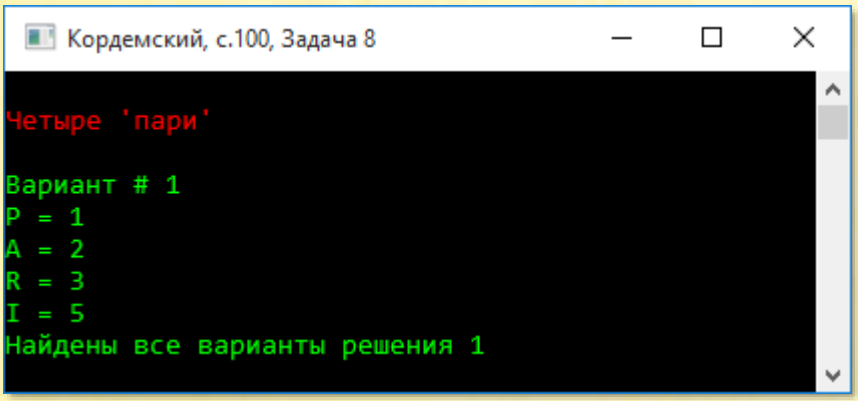
```
   Console.WriteLine('Четыре ''пари''');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                        nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Однако теперь нам предстоит проверить выполнение не одного, а сразу четырёх условий! Лучше их проверять последовательно, и если какое-либо условие окажется невыполненным, то с помощью оператора *continue* сразу переходить к проверке следующих значений переменных цикла P,A,R,I:

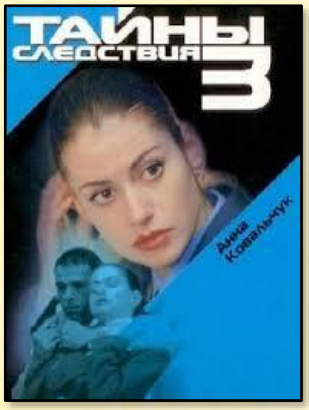
```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   Result := 0;
   for var P := 0 to 9 do
      for var A := 0 to 9 do
      begin
         if (A = P) then continue;
         for var R := 0 to 9 do
         begin
            if ((R = A) or (R = P)) then
               continue;
            for var I := 0 to 9 do
            begin
               if ((I = P) or (I = A) or (I = R)) then
                  continue;
               if (P * A * R * I <> 5 * (P + A + R)) then
                  continue;
               if (P * A * R * I <> 3 * (I + R + A)) then
                  continue;
               if (3 * P * A * R * I <> 10 * (P + I + R)) then
                  continue;
               if (4 * P * A * R * I <> 15 * (P + A + I)) then
                  continue;
```

```
                    Result += 1;
                    Console.WriteLine('Вариант # ' +
                                      result.ToString());
                    var s := 'P = ' + P;
                    Console.WriteLine(s);
                    s := 'A = ' + A;
                    Console.WriteLine(s);
                    s := 'R = ' + R;
                    Console.WriteLine(s);
                    s := 'I = ' + I;
                    Console.WriteLine(s);

                end
            end
        end
end;
```

Когда все равенства станут верными, мы печатаем решение задачи на экране:



Тайна трёх слагаемых



Функция без параметров
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор ***exit***
Оператор ***or***

Задача 11 из книги *Удивительный мир чисел* [КА96], страница 79:

Переведите на язык арифметики чисел:

```
  Б О Р Я
+   И Д И
  Б У Д Ь
---------
  Д О Б Р
```

Первое слагаемое (**БОРЯ**) должно быть возможно **максимальным**.

Этот ребус с *БОРЕЙ* – вполне **классический**: все цифры заменены буквами, и нужно узнать сумму нескольких чисел. Вы найдёте в книгах огромное множество таких ребусов, и все они решаются одинаково – с помощью вложенных циклов ***for***.

Решая классические ребусы, вы должны учитывать, что первая цифра числа не может быть нулём. Это значит, что буквы **Б**, **И**, **Д** могут принимать значения от 1 до 9, но не нуль. Это требование легко учесть в циклах *for* для соответствующих переменных, что мы и сделали в функции **Solve**:

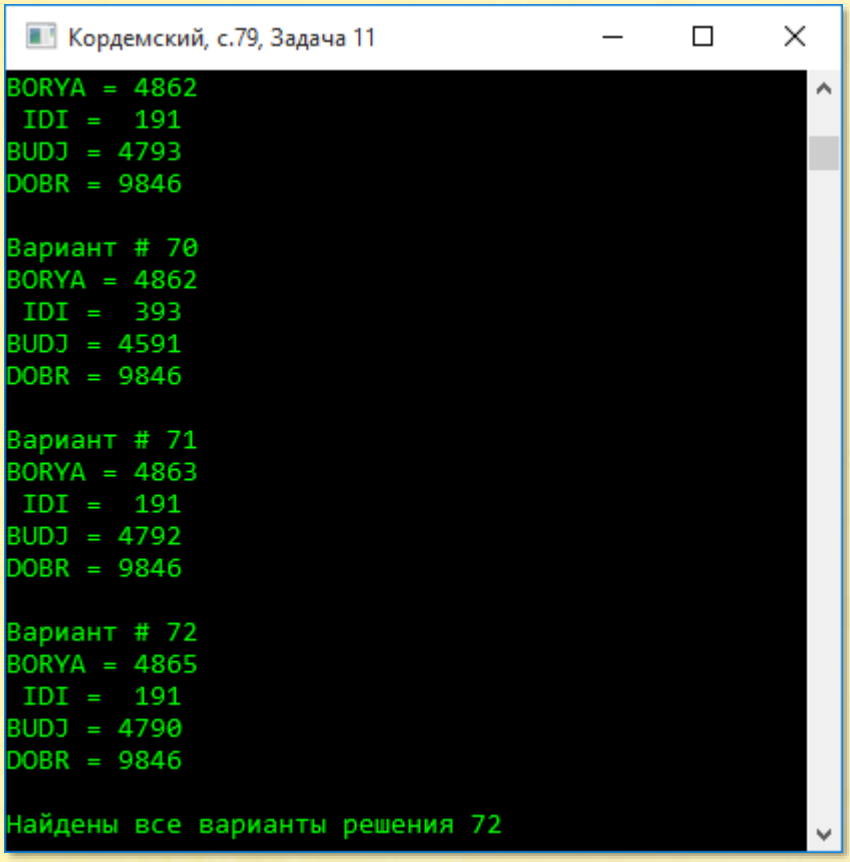
```
uses
   System;

// Кордемский, с.79, Задача 11
```

```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   Result := 0;
   for var B := 1 to 9 do
      for var O := 0 to 9 do
      begin
         if (O = B) then continue;
         for var R := 0 to 9 do
         begin
            if ((R = B) or (R = O)) then continue;
            for var YA := 0 to 9 do
            begin
               if ((YA = B) or (YA = O) or (YA = R)) then
                  continue;

               for var I := 1 to 9 do
               begin
                  if ((I = B) or (I = O) or (I = R) or
                                          (I = YA)) then
                     continue;
                  for var D := 1 to 9 do
                  begin
                     if ((D = B) or (D = O) or (D = R) or
                                   (D = YA) or (D = I)) then
                        continue;
                     for var U := 0 to 9 do
                     begin
                        if ((U = B) or (U = O) or
                            (U = R) or (U = YA) or
                                    (U = I) or (U = D)) then
                           continue;
                        for var J := 0 to 9 do
                        begin
                           if ((J = B) or (J = O) or
                               (J = R) or (J = YA) or
                               (J = I) or (J = D) or
                                                (J = U)) then
                              continue;
                           if (B * 1000 + O * 100 +
                               R * 10 + YA + I * 100 + D * 10 + I +
                               B * 1000 + U * 100 + D * 10 + J <>
                              D * 1000 + O * 100 + B * 10 + R) then
```

```
                            continue;

                        Result += 1;
                        Console.WriteLine('Вариант # ' +
                                          Result.ToString());
                        var s := 'BORYA = ' + B + O + R + YA;
                        Console.WriteLine(s);
                        s := ' IDI =  ' + I + D + I;
                        Console.WriteLine(s);
                        s := 'BUDJ = ' + B + U + D + J;
                        Console.WriteLine(s);
                        s := 'DOBR = ' + D + O + B + R;
                        Console.WriteLine(s);

                        Console.WriteLine();

                    end
                end
            end
        end
    end
end;


begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.79, Задача 11';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Тайна трёх слагаемых');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                     nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Кроме большого числа циклов и длинных проверок, ничего сложного в подобного рода задачах нет, поэтому вы решите любую из них без проблем!

Рисунок показывает, что найдено **72 ответа**:



Обычно хороший числобус должен иметь **единственное** решение, что в этой задаче достигается дополнительным требованием, чтобы число *БОРЯ* было максимальным.

Конечно, можно просто вручную найти в списке решений то, в котором это условие выполняется, но всякий перебор лучше перепоручать компьютеру.

Введём переменную **max**, которая поначалу имеет нулевое значение (или любое отрицательное):

```
var max := 0;
```

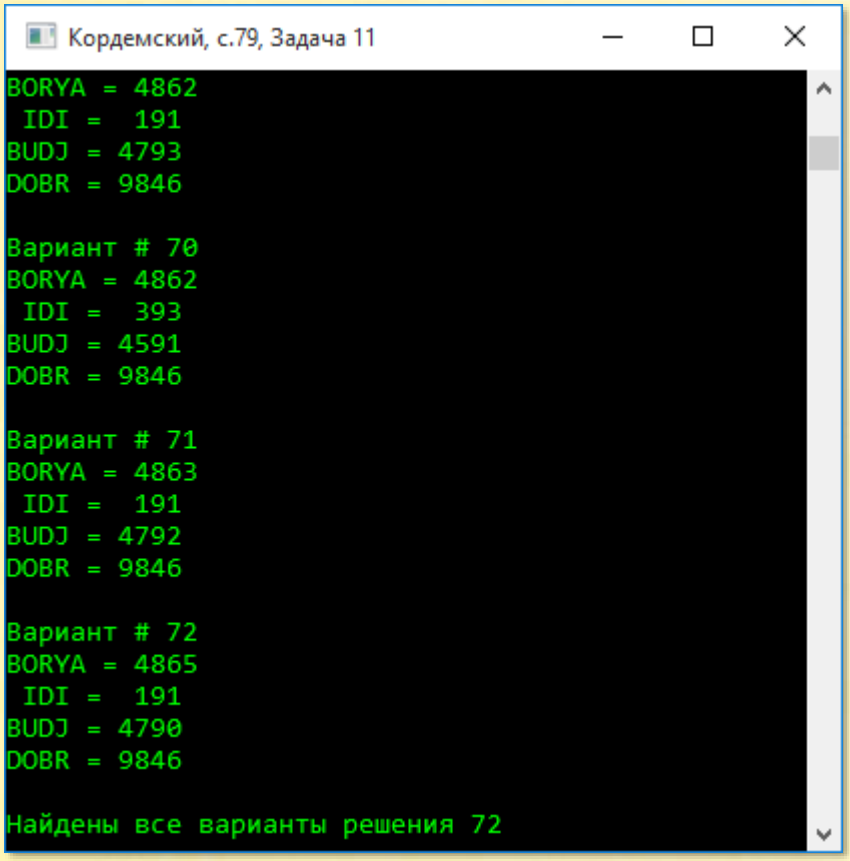
Перед строчкой

```
Result += 1;
```

вставляем код для проверки и запоминания максимального значения для *БОРИ*:

```
if (max > B*1000 + O*100 + R*10 + YA) then
    continue;
 // запоминаем новое максимальное значение:
 max := B * 1000 + O * 100 + R * 10 + YA;

 Result += 1;
```

Теперь функция *Solve* будет печатать только «рекордные» результаты для числа *БОРЯ*. Самое большое значение будет последним в списке.



На рисунке вы видите любопытную картину – число *БОРЯ* увеличивается с каждой итерацией и достигает своего максимума только в самом конце списка решений:

Мы можем предположить, что автор задачи решал её так же, как и мы, то есть на компьютере.

Меняем четыре буквы на четыре цифры

Функция без параметров
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор ***exit***
Оператор ***or***

Задача 10 из книги *Удивительный мир чисел* [КА86], страница 73:

В равенстве

$7 \times (a2 + b)^3 = 23c2k$ (1)

и независимо от него в равенстве

$37037 \times ab = 19019 \times ck$ (2)

замените буквы a, b, c, k на подходящие цифры так, чтобы оба равенства подтвердились.

Очень простая задача – всего на 4 вложенных цикла!

```
uses
   System;

// Кордемский, с.73, Задача 10

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.73, Задача 10';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Меняем четыре буквы на четыре цифры');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();
```

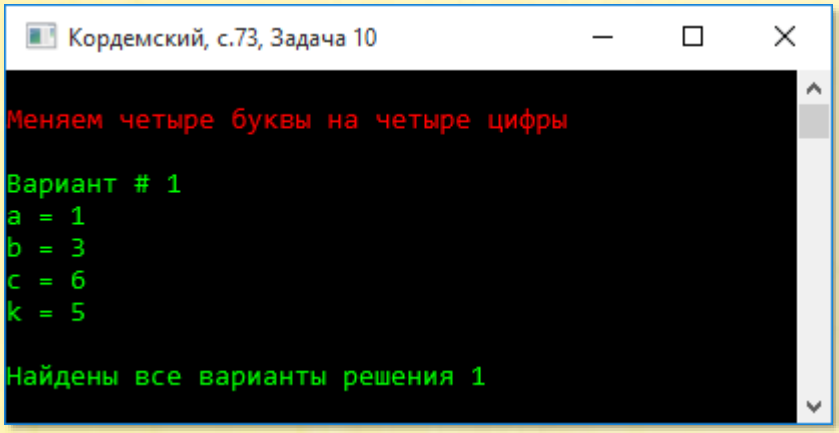
```
   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                        nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

В функции Solve, пожалуй, стоит обратить внимание только на выделенные строчки, в которых проверяется условие первой задачи:

```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   Result := 0;
   for var a := 0 to 9 do
      for var b := 0 to 9 do
      begin
         if (b = a) then continue;
         for var c := 0 to 9 do
         begin
            if ((c = a) or (c = b)) then continue;
            for var k := 0 to 9 do
            begin
               if ((k = a) or (k = b) or (k = c)) then
                  continue;
               var a2b := a * 10 + 2 + b;
               if (7 * a2b * a2b * a2b <> 2 * 10000 + 3 * 1000 +
                              c * 100 + 20 + k) then
                  continue;

               Result += 1;
               Console.WriteLine('Вариант # ' +
                                    Result.ToString());
               var s := 'a = ' + a;
               Console.WriteLine(s);
               s := 'b = ' + b;
               Console.WriteLine(s);
               s := 'c = ' + c;
               Console.WriteLine(s);
               s := 'k = ' + k;
```

```
                Console.WriteLine(s);

                Console.WriteLine();
            end
        end
    end
end;
```

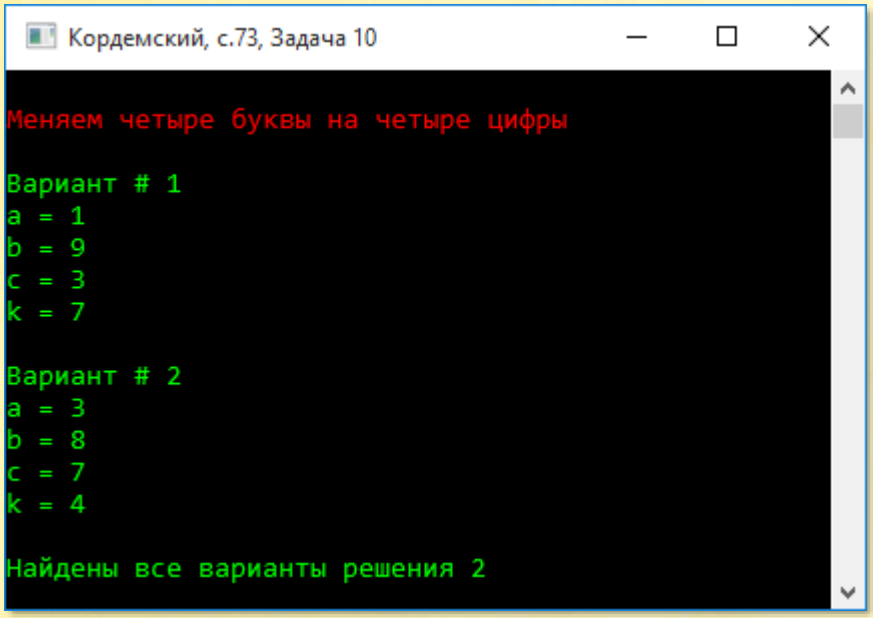
Первая часть задачи решена:



Чтобы решить вторую часть задачи, следует изменить условие:

```
//var a2b := a * 10 + 2 + b;
//if (7 * a2b * a2b * a2b <> 2 * 10000 + 3 * 1000 +
//              c * 100 + 20 + k) then
//   continue;
var ab := a * 10 + b;
var ck := c * 10 + k;
if (37037 * ab <> 19019 * ck) then
   continue;
```

Вторая часть задачи имеет 2 решения:



Коварная задача папы

Функция без параметров
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор ***exit***
Оператор ***or***

Задача 11 из книги *Удивительный мир чисел* [КА86], страницы 22-23:

Мой папа - шофёр - часто ездит по трассе, вдоль которой расположены пять небольших поселков: A, B, C, D, E. Папа знал точно, сколько домов в каждом из этих посёлков, и составил для меня такую задачу: «Сколько всего домов в этих пяти посёлках, если

- в A и B вместе 13 домов,

- в B и C вместе 31 дом,
- в C и E - 17 домов,
- в E и D - 26 домов,
- в A и D - 23 дома?»

Я подметил, что число домов в каждом посёлке подсчитывалось папой дважды (по тексту задачи), поэтому моё решение было молниеносным:

(13 + 31 + 17 +26+ 23) / 2 = 55 (домов в пяти поселках вместе).

Но... вот уж действительно: поспешишь — людей насмешишь! Оказалось, что задачу папа сознательно составил так, что она не может быть решена. И я должен был это доказать. Помогите!

В каждом посёлке не меньше 0 домов и не больше 31 дома – эти числа легко получить из условия задачи. Достаточно перебрать все варианты числа домов в этом диапазоне для посёлков A, B, C, D, E, чтобы найти все решения задачи.

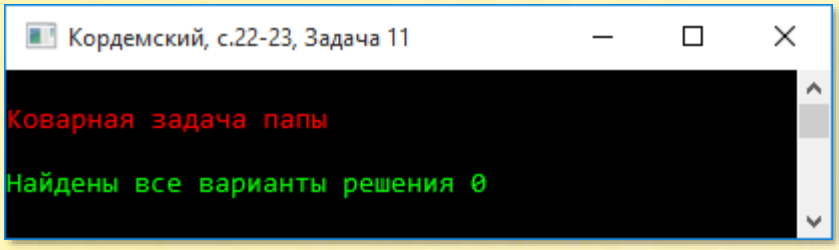
```
uses
   System;

// Кордемский, с.22-23, Задача 11

begin
   //заголовок окна:
   Console.Title := 'Кордемский, с.22-23, Задача 11';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Коварная задача папы');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                        nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Выписываем 5 вложенных циклов *for* и по ходу итераций проверяем папины фантазии:

```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   Console.ForegroundColor := ConsoleColor.Yellow;
   Result := 0;
   var min := 0;
   var max := 31;
   for var A := min to max do
      for var B := min to max do
      begin
         if (A + B <> 13) then continue;
         for var C := min to max do
         begin
            if (B + C <> 31) then continue;
            for var D := min to max do
            begin
               if (A + D <> 23) then continue;
               for var E := min to max do
               begin
                  if (C + E <> 17) then continue;
                  if (E + D <> 26) then continue;

                  Result += 1;
                  Console.WriteLine('Вариант # ' +
                                    Result.ToString());
                  var s := 'A = ' + A;
                  Console.WriteLine(s);
                  s := 'B = ' + B;
                  Console.WriteLine(s);
                  s := 'C = ' + C;
                  Console.WriteLine(s);
                  s := 'D = ' + D;
                  Console.WriteLine(s);
                  s := 'E = ' + E;
                  Console.WriteLine(s);

                  Console.WriteLine();
               end
            end
```

```
            end
        end
end;
```

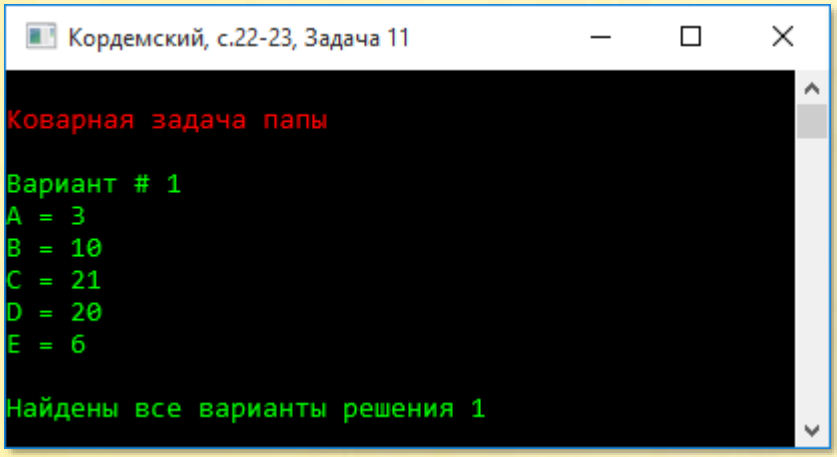
Как и отметил сынуля, папа его дезинформировал:



Правда, если в условии задачи нежно подправить условие:

```
C + E = 17 → C + E = 27
```

то папина задача (а я подозреваю, что это был либо папа Карло, либо Валдис Пельш) успешно разрешится *единственным* способом:



На ферме

Процедура без параметров
Константы
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***

Задача 12 (13) из книги *Удивительный мир чисел* [КА86], страница 53:

На ферме выращивают кроликов и фазанов. В настоящее время их столько, что у всех вместе 740 голов и 1980 ног.

Сколько же в настоящее время находится на ферме кроликов и фазанов?

```
uses System;

// Кордемский, с.53, Задача 12


begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.53. Задача 12';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('На ферме');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

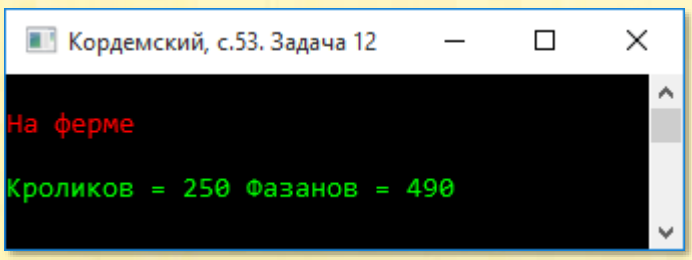
Поскольку и число голов, и число ног у кроликов и фазанов выражается **целыми** числами, то достаточно перебрать все варианты распределения 740 голов по кроликам и фазанам:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
  const GOLOVY = 740;
  const NOGI = 1980;
begin
  for var kroliki := 0 to GOLOVY do
  begin
     for var fazany := 0 to GOLOVY do
       begin
            if (kroliki + fazany <> GOLOVY) then
               continue;
            if (kroliki*4 + fazany*2 <> NOGI) then
               continue;
```

При этом мы должны учитывать, что у кроликов по 4 ноги, а у фазанов – только по 2:

```
            Console.WriteLine('Кроликов = {0} Фазанов = {1}',
                              kroliki, fazany);
            Console.WriteLine();
       end
   end
end;
```

И вмиг кролики и фазаны пересчитаны и занесены в *Консольное окно* нашей программы:



Решите систему уравнений

Функция без параметров
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор ***exit***

Задача 7 из книги *Удивительный мир чисел* [КА86], страница 100:

Решите систему уравнений:

$$\begin{cases} \sqrt[x]{x+y} = 2, \\ (x+y)\cdot 6^x = 248\,832. \end{cases}$$

Далеко не очевидно, но значения переменных *x* и *y* должны быть **целыми**, иначе верхнее равенство вряд ли выполнится. Его лучше записать в другом виде:

$$x + y = 2^x$$

```
uses
   System;

// Кордемский, с.100, Задача 7

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.100. Задача 7';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
```

```
   Console.WriteLine('Решите систему уравнений');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения: ' +
                     nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

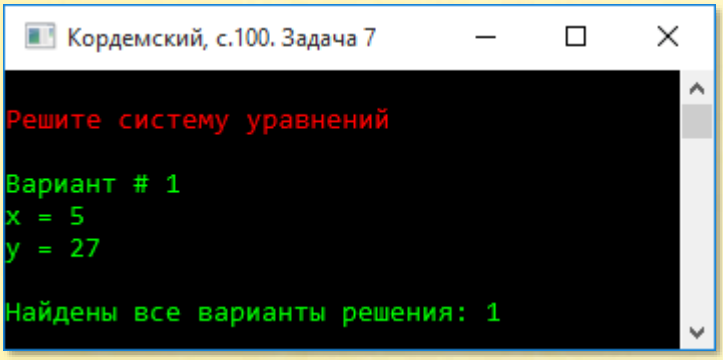
Пределы изменения переменных *x* и *y* можно взять достаточно большими, но если решение не будет найдено, то верхнюю границу следует ещё увеличить.

Больше никаких трудностей в решении задачи не наблюдается, поэтому в двух вложенных циклах *for* мы перебираем значения переменных *x* и *y* и проверяем выполнение условий задачи:

```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   Result := 0;
   for var x := 0 to 100 do
   begin
      for var y := 0 to 100 do
      begin
         if (x + y <> Math.Pow(2, x)) then
            continue;
         if ((x + y) * Math.Pow(6, x) <> 248832) then
            continue;

         Result += 1;
         Console.WriteLine('Вариант # ' +
                           Result.ToString());
         var s := 'x = ' + x;
         Console.WriteLine(s);
         s := 'y = ' + y;
```

```
            Console.WriteLine(s);
            Console.WriteLine();
        end
    end
end;
```

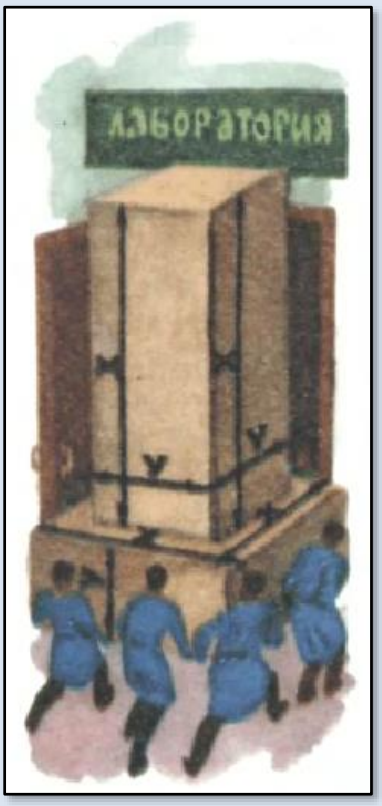
На рисунке вы видите ответ на задачу. Значения переменных оказались совсем небольшими!



Сооружение для лаборатории

Функция без параметров
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор ***exit***

Задача 7 из книги *Удивительный мир чисел* [КА86], страница 64:



Для одной лаборатории изготовили конструкцию из двух спаянных пустотелых параллелепипедов (см. рисунок справа). Длины всех рёбер (в дециметрах) - целые числа. Основания параллелепипедов — квадраты. Высота первого параллелепипеда равна стороне основания второго, а высота второго равна стороне основания первого параллелепипеда.

Поместится ли эта конструкция, объём которой 3900 дм^3, в лаборатории, если расстояние от пола лаборатории до потолка равно 2 м 75 см?

```
uses
   System;

// Кордемский, с.64, Задача 7

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.64. Задача 7';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Сооружение для лаборатории');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения: ' +
                     nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Поскольку *x* и *y* – целые числа, то мы можем просто перебрать все их возможные комбинации. Например, можно сразу заключить, что оба эти числа больше нуля.

Из уравнения

$$x^2*y + x*y^2 = 3900 \quad (1)$$

следует, что ни одно из этих чисел не больше корня квадратного из 3900. Точное значение корня нам не нужно, пусть это будет 70.

Осталось в двух вложенных циклах *for* составить все пары чисел *x* и *y* и для каждой пары проверить выполнение условия (1):

```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   Result := 0;
   for var x := 1 to 70 do
   begin
      for var y := 1 to 70 do
      begin
         if (x * x * y + x * y * y <> 3900)   then
            continue;
```

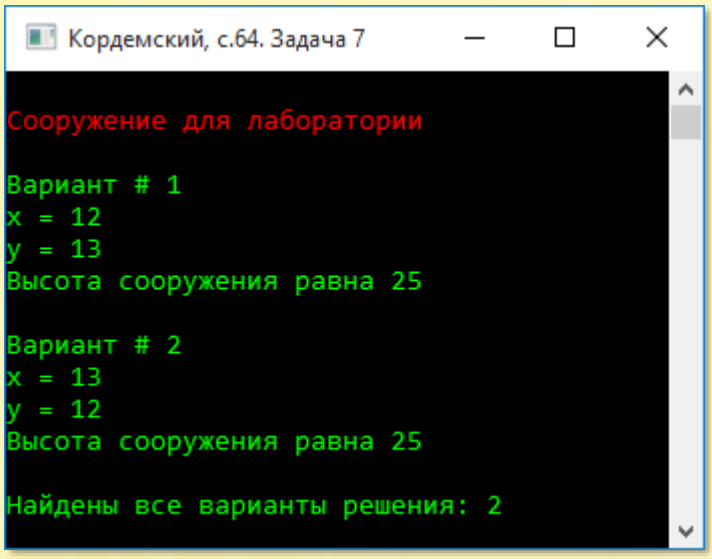
Если оно выполняется, значит, размеры конструкции найдены:

```
Result += 1;
Console.WriteLine('Вариант # ' +
                  Result.ToString());
var s := 'x = ' + x;
Console.WriteLine(s);
s := 'y = ' + y;
Console.WriteLine(s);
```

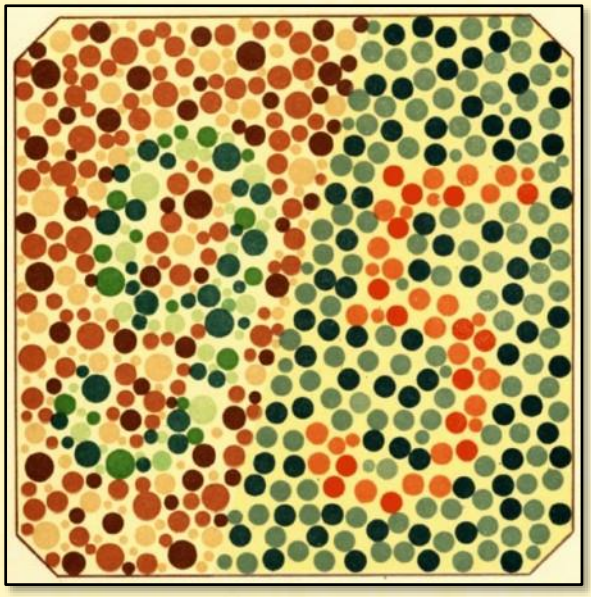
Общая высота сооружения равна *x+y*. Значение этой суммы мы печатаем на экране:

```
            s := 'Высота сооружения равна ' + (x + y);
            Console.WriteLine(s);
            Console.WriteLine();
        end
    end;
end;
```

Как показывает рисунок, либо x = 12, y = 13, либо y = 12, x = 13, но в обоих случаях высота конструкции составляет 12 + 13 = 25 дециметров = 2,5 метра, что меньше высоты лаборатории. Значит, конструкция в ней поместится.



И такие есть числа



Функция без параметров
Цикл for
Условный оператор if
Оператор continue
Оператор exit

Задача 4-1 из книги *Удивительный мир чисел* [КА86], страница 63:

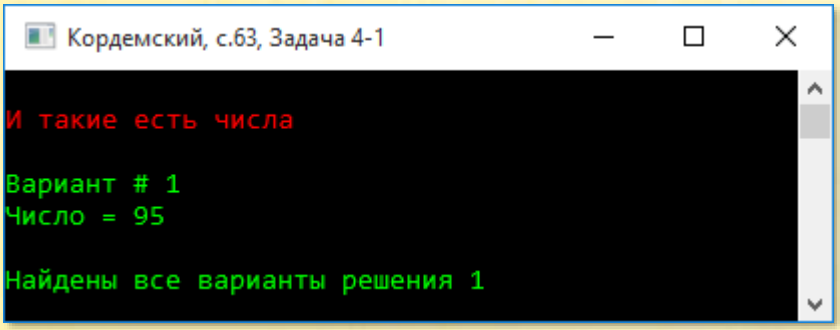
Какое двузначное число в 19 раз больше числа его единиц?

```
uses
   System;

// Кордемский, с.63, Задача 4-1

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.63, Задача 4-1';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('И такие есть числа');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                        nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Чтобы решить задачу, достаточно перебрать все двузначные числа и проверить условие задачи:

```
// РЕШАЕМ ЗАДАЧУ
function Solve() : integer;
begin
    Result := 0;
    for var n := 10 to 99 do
    begin
        if (n <> n mod 10*19) then
            continue;

      Result += 1;
      Console.WriteLine('Вариант # ' +
                        result.ToString());
        var s := 'Число = ' + n;
        Console.WriteLine(s);
        Console.WriteLine();
   end;
end;
```

Как вы видите на рисунке, задача имеет *единственное* решение: искомое число равно 95:



И такие есть числа 3

Процедура без параметров
Вложенные циклы
Условный оператор ***if***
Оператор ***continue***
Функция *с параметром*
Оператор ***exit***
Функция ***Trunc***
Оператор целочисленного деления ***mod***
Оператор ***and***

Задача 4-3 из книги *Удивительный мир чисел* [КА86], страница 63:

> Два простых числа обладают свойством: если от каждого из них вычесть половину другого, то одна разность будет в 5 раз больше другой.
>
> Найдите эти числа.

```
uses
   System;

// Кордемский, с.63. Задача 4.3

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.63. Задача 4.3';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('И такие есть числа');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Пусть первое число **n1** меньше второго числа **n2**. Тогда *n2* как минимум на 1 больше *n1*, но не более, чем в 2 раза.

Первое число мы ищем в бесконечном цикле ***while***, начиная с двойки - первого простого числа. Второе число ищем в бесконечном цикле ***for***, начиная со следующего числа и заканчивая числом, вдвое большим первого:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var n1 := 2;
   while (true) do
   begin
      for var n2 := n1 + 2 to 2 * n1 do
      begin
```

Для каждой пары чисел проверяем **условие** задачи:

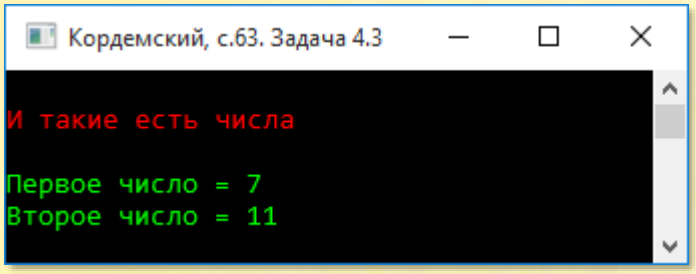
```
if (2 * n2 - n1 <> 5 * (2 * n1 - n2)) then
    continue;
```

Чтобы избежать дробных чисел, мы оба числа умножили на двойку.

Как только оно выполнится, мы **печатаем** решение на экране и заканчиваем поиски:

```
         var s := 'Первое число = ' + n1;
         Console.WriteLine(s);
         s := 'Второе число = ' + n2;
         Console.WriteLine(s);
         Console.WriteLine();
         exit;
      end;
      n1 += 1;
   end
end;
```

Обратите внимание, что для решения этой задачи нам даже не потребовалось проверять числа *n1* и *n2* на простоту:



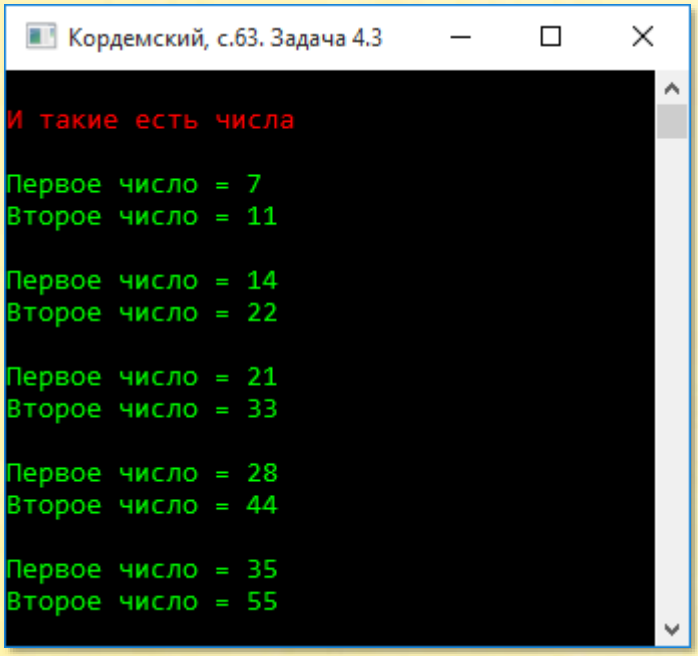
Но если вы закомментируете строку:

```
//              exit;
```

то найдёте ещё множество пар чисел, которые удовлетворяют условиям задачи:

> *если от каждого из них вычесть половину другого, то одна разность будет в 5 раз больше другой.*

Но при этом не являются простыми:



Подкорректируем процедуру Solve:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var n1 := 2-1;
   while (true) do
   begin
      n1 += 1;
      if (not IsPrime(n1)) then
         continue;
      for var n2 := n1 + 2 to 2 * n1 do
      begin
         if (not IsPrime(n2)) then
            continue;
         if (2 * n2 - n1 <> 5 * (2 * n1 - n2)) then
            continue;

         var s := 'Первое число = ' + n1;
         Console.WriteLine(s);
         s := 'Второе число = ' + n2;
         Console.WriteLine(s);
         Console.WriteLine();
         exit;
      end;
      //if n1 > 35 then exit;
   end
end;
```

Теперь каждое число *n1* и *n2* мы дополнительно проверяем на простоту с помощью функции IsPrime:

```
// ОПРЕДЕЛЯЕМ ЯВЛЯЕТСЯ ЛИ ЗАДАННОЕ ЧИСЛО
// ПРОСТЫМ
function IsPrime(num: int64): boolean;
begin
   if ((num > 2) and (num mod 2 = 0)) then
   begin
      Result := false;
      exit;
```

```
    end;

    var endnum := Trunc(Math.Sqrt(num));
    var i := 3;
    while (i < endnum) do
    begin
      if (num mod i = 0) then
      begin
        Result := false;
        exit;
      end;
      i += 2;
    end;
    Result := true;
end;
```

Ответ на задачу мы получим тот же самый, но процедура *Solve* теперь «правильная».

И такие есть числа 4

Процедура без параметров
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор ***or***

Задача 4-4 из книги *Удивительный мир чисел* [КА86], страница 63:

> Сумма первого числа и квадрата второго равна 57, а сумма второго числа и квадрата первого равна 71.
>
> Найдите эти числа.

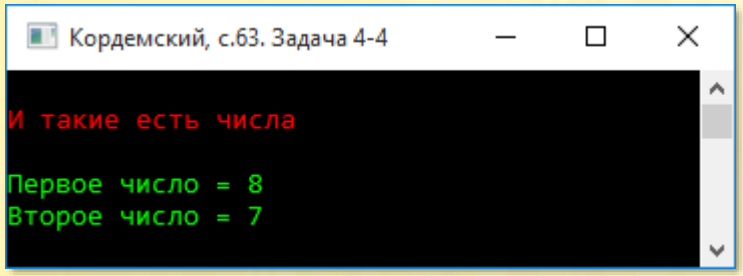
```
uses System;

// Кордемский, с.63, Задача 4-4

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.63. Задача 4-4';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('И такие есть числа');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Из условия задачи следует, что оба числа не меньше 1 и не больше 9. Этого вполне достаточно, чтобы во вложенных циклах *for* проверить все возможные пары чисел:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
 begin
 for var n1 := 1 to 9 do
    begin
        for var n2 := 1 to  9 do
        begin
             if ((n1 + n2*n2 <> 57) or (n1*n1 + n2 <> 71)) then
                  continue;

             var s := 'Первое число = ' + n1;
             Console.WriteLine(s);
             s := 'Второе число = ' + n2;
             Console.WriteLine(s);
             Console.WriteLine();
        end
    end
```

```
end;
```

Перебор оказался небольшим, поэтому ответ получаем практически мгновенно:



На базаре

Процедура без параметров
Константы
Вложенные циклы for
Условный оператор if
Оператор continue
Форматированный вывод

Задача 7 (2.3) из книги *Удивительный мир чисел* [КА86], страница 51:

На базаре за 9 кг орехов и 2 кг фейхоа заплатили столько же денег, сколько за 6 кг гранатов. А за 6 кг орехов, 5 кг фейхоа и 4 кг гранатов заплатили 43 р.

Сколько стоит 1 кг орехов, фейхоа и гранатов отдельно, если известно, что стоимость каждого продукта выражается целым числом рублей?

```
uses
   System;

// Кордемский, с.51, Задача 7
```

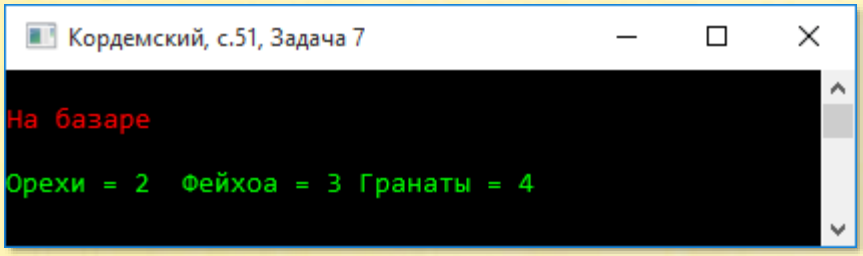
```
begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.51, Задача 7';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('На базаре');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Эта базарная задача легко решается простым перебором:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
const
   COST = 43;
   COREHI = COST div 6;
   CFEIHOA = COST div 5;
   CGRANAT = COST div 4;

begin
   for var orehi := 0 to COREHI do
   begin
      for var feihoa := 0 to CFEIHOA do
      begin
         for var granat := 0 to CGRANAT do
         begin
            if (orehi * 6 + feihoa * 5 + granat * 4 <> COST) then
               continue;

            if (orehi * 9 + feihoa * 2 <> granat * 6)    then
               continue;

            Console.WriteLine('Орехи = {0}  Фейхоа = {1}
                               Гранаты = {2}', orehi, feihoa, granat);
```

```
            Console.WriteLine();
        end
    end
  end
end;
```

Теперь вы можете смело отправляться на рынок 1986 года:



Дедушка и внучка

Процедура без параметров
Цикл repeat-until
Форматированный вывод
Оператор деления div

Задача 10 (10.1) из книги *Удивительный мир чисел* [КА86], страница 52:

Сколько дедушке лет, столько месяцев внучке. Дедушке с внучкой вместе 91 год.

Сколько лет дедушке и сколько внучке?

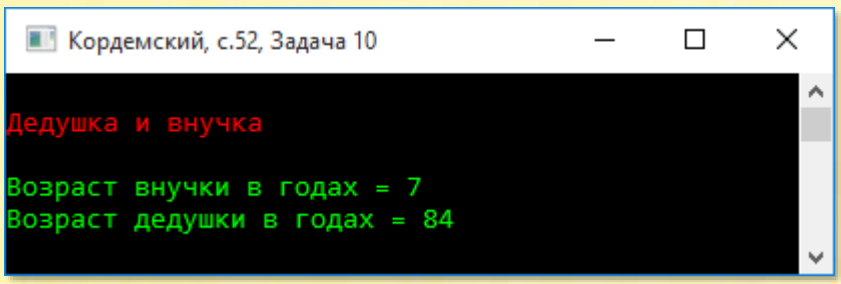
```
uses
   System;

// Кордемский, с.52, Задача 10
```

```
begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.52, Задача 10';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Дедушка и внучка');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Самое сложное в этой задаче – сформулировать **проверку** в операторе *until*:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   // возраст внучки в месяцах:
   var x := 0;

   repeat
      x += 1;
   until not (x + 12 * x <> 91 * 12);

   Console.WriteLine('Возраст внучки в годах = {0}', x div 12);
   Console.WriteLine('Возраст дедушки в годах = {0}', 91 - x div 12);

   Console.WriteLine();
end;
```

Обозначив через *x* возраст внучки в месяцах, мы найдём возраст дедушки в годах – тоже *x*, а в месяцах – *12x*. Дальше задача решается очень просто:



Сколько у мамы дочерей и сыновей?

Процедура без параметров
Константы
Вложенные циклы **for**
Условный оператор **if**
Оператор **continue**
Оператор деления **/**
Форматированный вывод

Это и есть фейхоа!

Задача 15 (4) из книги *Удивительный мир чисел* [КА86], страница 55:

> В семье 12 детей. Мама принесла для них 70 штук фейхоа. Половину всех фейхоа она раздала дочерям поровну, остальные отдала сыновьям, которые разделили их между собой также поровну. Каждый мальчик получил на 2 фейхоа больше, чем каждая девочка.
>
> Сколько было у этой мамы дочерей и сыновей?

С точки зрения математики, мальчики и девочки ничем не отличаются от кроликов и фазанов, поэтому эту задачу мы решаем так же, как и про животных на ферме:

```
uses
   System;

// Кордемский, с.55, Задача 15

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.55, Задача 15';
   Console.WriteLine('');
```

```
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Сколько у мамы дочерей и сыновей?');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
const
   // число фейхоа:
   FEIJOA = 70;
   // число детей:
   CHILDREN = 12;

begin
   // число мальчиков в семье:
   var m := 0;
   // число девочек в семье:
   var d := 0;

   for m := 1 to CHILDREN do
   begin
      for d := 1 to CHILDREN do
      begin
         if (m + d <> CHILDREN) then
            continue;
```

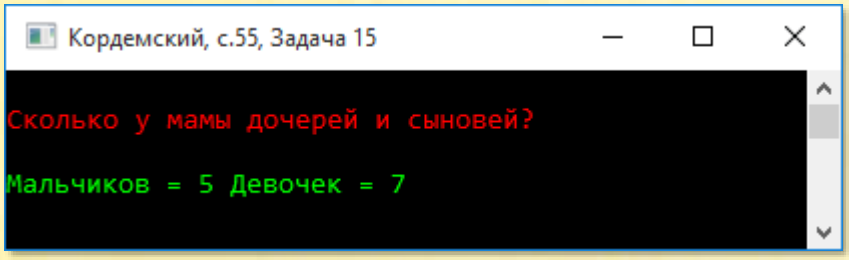
Единственная закавыка этой задачи - в написании вот этой проверки:

```
if (FEIJOA / 2 / m - FEIJOA / 2 / d <> 2) then
    continue;
```

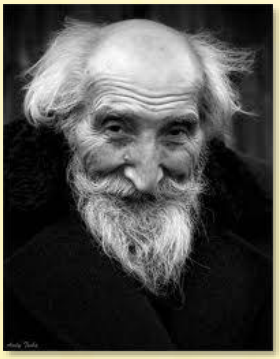
Впрочем, при внимательном чтении условия задачи легко сообразить, что девочки и мальчики получили странных фруктов фейхоа поровну, то есть по FEIJOA/2 штук. Всё остальное – дело техники, то есть компьютера.

```
            Console.WriteLine('Мальчиков = {0} Девочек = {1}',
                              m, d);
            Console.WriteLine();
        end
    end;
    Console.WriteLine();
end;
```

А на рисунке представлен ответ на задачу:



Из жизни Дефурнеля



Процедура без параметров
Цикл while
Комбинированные операторы присваивания
Оператор and

Задача 24 (26) из книги *Удивительный мир чисел* [КА86], страница 57:

Некогда в отрывном календаре были приведены любопытные факты из биографии француза Пьера Дефурнеля. Он был отцом трёх сыновей, родившихся в

разных веках. Он сам и старший сын родились в XVII веке. Женился он на девятнадцатом году жизни. Через год стал отцом первого сына. Спустя 37 лет женился второй раз. Через год стал отцом второго сына. А ещё через 43 года родилась его будущая третья жена. Через 19 лет после этого они поженились. Через год родился третий сын. Спустя
8 лет Пьер Дефурнель умер.

Установите годы рождения Пьера Дефурнеля, сыновей, третьей жены и годы, когда женился Дефурнель и когда он умер?

```
uses
   System;

// Кордемский, с.57, Задача 24

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.57, Задача 24';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Из жизни Дефурнеля');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Поскольку XVII век начинается с 1601 года, то мы можем в цикле ***while*** приступить к перебору с предыдущего года и добавлять по 1 к текущему числу **year** – году рождения Дефурнеля:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   // год рождения Дефурнеля:
```

```
var year := 1600;
// годы рождения сыновей:
var son1 := 0;
var son2 := 0;
var son3 := 0;

var flg := false;
while (not flg) do
begin
   year += 1;
```

И так мы продолжаем итерации до тех пор, пока не выполнятся все условия задачи касательно года рождения каждого из трёх его сыновей. Все условия легко формируются при внимательном чтении условия задачи:

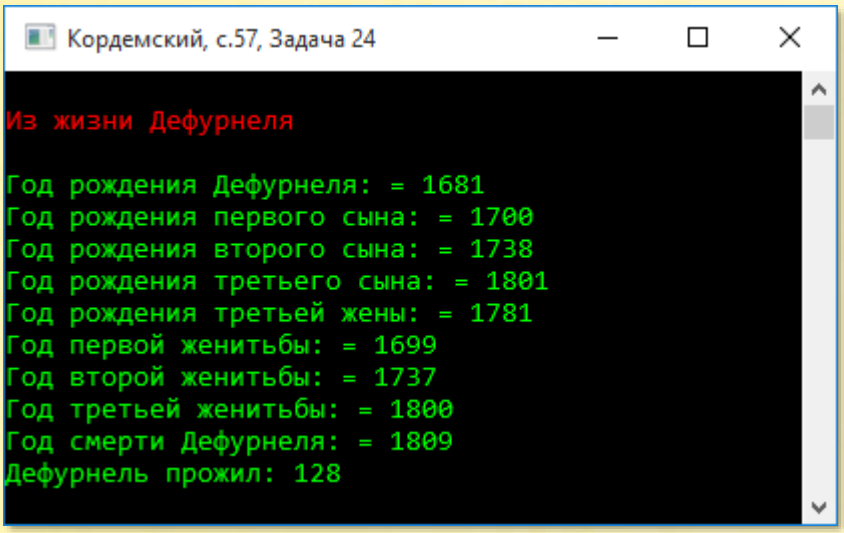
```
   // год рождения первого сына:
   son1 := year + 19;
   flg := son1 < 1701;
   // год рождения второго сына:
   son2 := son1 + 38;
   flg := flg and (son2 >= 1701) and (son2 < 1801);
   // год рождения третьего сына:
   son3 := son2 + 63;
   flg := flg and (son3 >= 1801) and (son3 < 1901);
end;
```

Когда год рождения Дефурнеля будет установлен, мы печатаем полезную и занимательную статистическую информацию на экране:

```
Console.WriteLine('Год рождения Дефурнеля: = {0}', year);
Console.WriteLine('Год рождения первого сына: = {0}', son1);
Console.WriteLine('Год рождения второго сына: = {0}', son2);
Console.WriteLine('Год рождения третьего сына: = {0}', son3);
Console.WriteLine('Год рождения третьей жены: = {0}', son2 + 43);
Console.WriteLine('Год первой женитьбы: = {0}', son1 - 1);
Console.WriteLine('Год второй женитьбы: = {0}', son2 - 1);
Console.WriteLine('Год третьей женитьбы: = {0}', son3 - 1);
Console.WriteLine('Год смерти Дефурнеля: = {0}', son3 + 8);
Console.WriteLine('Дефурнель прожил: {0}', son3 + 8 - year);
```

```
   Console.WriteLine();
end;
```

Она представленна в полном объёме на рисунке:



Первый числовой фокус

Функция ***ReadInteger***
Оператор ***exit***
Условный оператор ***if***
Процедура *с параметром*
Оператор деления по модулю ***mod***
Оператор деления ***div***
Операторы ***< > or <>***

А вот **фокус** из книги *Удивительный мир чисел* [КА86], страница 34, задача 2:

> Скажите другу: «Любое трёхзначное число умножь на 37, потом на 27. К полученному шестизначному числу прибавь удвоенное первоначальное число. Покажи мне результат, и я угадаю задуманное трёхзначное число».

Секрет фокуса. Пусть задумано трёхзначное число

100x + 10y + z,

где x, y, z - цифры сотен, десятков и единиц соответственно. Выполнив указанные действия, получим:

100 100x + 10 010y + 1001z= 1001 • (100x + 10y + z).

Теперь ясно, что достаточно разделить результат на 1001, чтобы получилось задуманное трёхзначное число.

Пример. Пусть задумано число 173. После выполнения указанных действий получилось 173 173. Наблюдаем, что в его записи трёхзначное число 173 повторяется. Вычеркнем 173; это равносильно тому, что 173 173 разделили на 1001; в результате остается задуманное число 173.

Замечание. При повторении фокуса легко обнаружится, что если задумано число *abc*, то результат, показываемый фокуснику, имеет вид *abcabc* . Чтобы это скрыть, надо добавить еще одно действие в конце фокуса, например потребовать прибавить 1111. Тогда фокуснику скажут не 173 173, а 174 284. Теперь закономерность скрыта, а фокуснику ничего не стоит в уме вычесть 1111, а затем угадать задуманное число.

В **главном блоке** программы мы предлагаем всем желающим загадать трёхзначное число и произвести с ним требуемые манипуляции:

```
uses
   System;

// Кордемский, страница 34, задача 2

begin
   // заголовок окна:
   Console.Title := 'Первый числовой фокус';
   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Первый числовой фокус');
```

```
   // бесконечный цикл ввода данных -
   // пока пользователь не закроет программу
   // или не введет 0:
   repeat
      Console.ForegroundColor := ConsoleColor.Green;
      Console.WriteLine();
      Console.WriteLine('Любое трёхзначное число умножь на 37,');
      Console.WriteLine('потом на 27. ');
      Console.WriteLine('К полученному шестизначному числу прибавь');
      Console.WriteLine('удвоенное первоначальное число. ');
      Console.WriteLine('Введи результат, ');
      Console.Write('и я угадаю  задуманное трёхзначное число.  >  ');
      var num := ReadInteger();
      if (num = 0) then exit;

      Solve(num);

      Console.WriteLine();
   until not (true);
end.
```

Сам фокусник притаился в процедуре Solve, которая получает от загадчика 6-значное число. Но он не шибко доверчив, поэтому, прежде всего, проверяет число загадчика и отказывается фокусничать, если ему пытаются подсунуть негодное число:

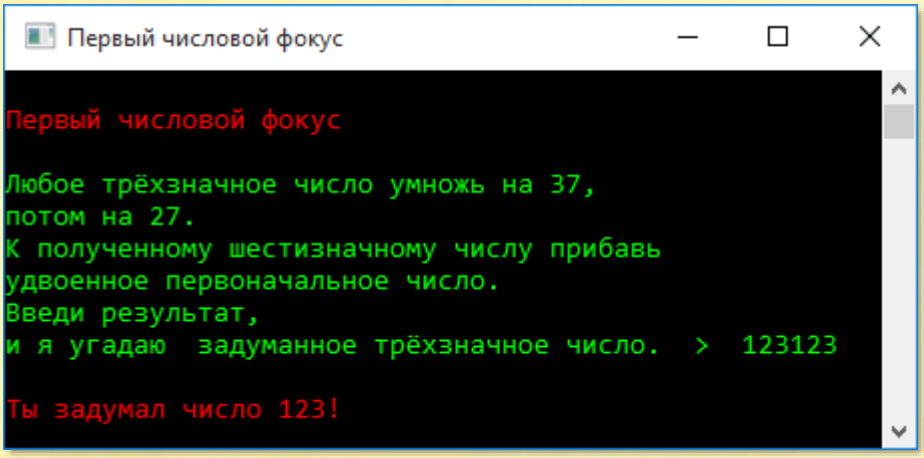
```
// ОТГАДЫВАЕМ ЧИСЛО
procedure Solve(num: integer);
const
   MIN6 = 100000;
   MAX6 = 999999;
begin
   Console.ForegroundColor := ConsoleColor.Red;
   if ((num < MIN6) or (num > MAX6)) then
   begin
      Console.WriteLine('Число должно быть шестизначным!');
      Console.WriteLine();
      exit;
   end;
   if (num mod 1001 <> 0) then
   begin
```

```
      Console.WriteLine('Ты ошибся в вычислениях!');
      Console.WriteLine();
      exit;
   end;
```

А добросовестному загадчику наш виртуальный фокусник безошибочно сообщает задуманное трёхзначное число:

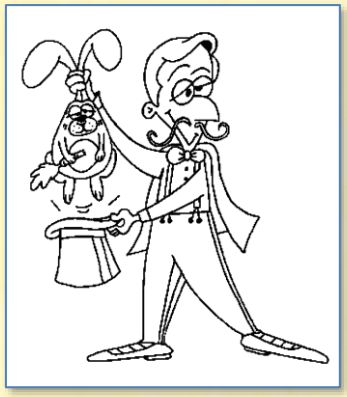
```
   Console.WriteLine();
   Console.WriteLine('Ты задумал число {0}!', num div 1001);
   Console.WriteLine();
end;
```

Например, моё число не поставило фокусника в тупик:



Фокус интересный, но очень простой. Постарайтесь учесть замечание и улучшить программу!

Второй числовой фокус



Цикл ***repeat-until***
Конкатенация ***строк***
Функция ***ReadInteger***
Условный оператор ***if***
Процедура *с параметром*
Оператор деления по модулю ***mod***
Оператор ***exit***

Второй **фокус** из книги *Удивительный мир чисел* [КА86], страница 35, задача 3 немного сложнее первого:

Двое участников задумывают одно трёхзначное число.

Первый умножает его на 27, потом на 37.
Второй умножает задуманное число на 13, потом умножает на 77.

Затем фокусник предлагает им сложить получившиеся два шестизначных числа и результат показать ему. Фокусник сообщает, какое трёхзначное число было задумано.

Секрет фокуса. Пусть задумано трёхзначное число

$100x + 10y + z$.

Выполнив указанные действия, получим:

$200\,000x + 20\,000y + 2000z = 2000 \cdot (100x + 10y + z)$.

Становится ясным, что достаточно разделить результат на 2000, чтобы получить задуманное число.

Сначала - для убедительности - проверим действие фокуса на контрольном примере:

```
123 * 27 * 37 =     122877
123 * 13 * 77 =     123123
                    246000
246000 : 2000 =        123
```

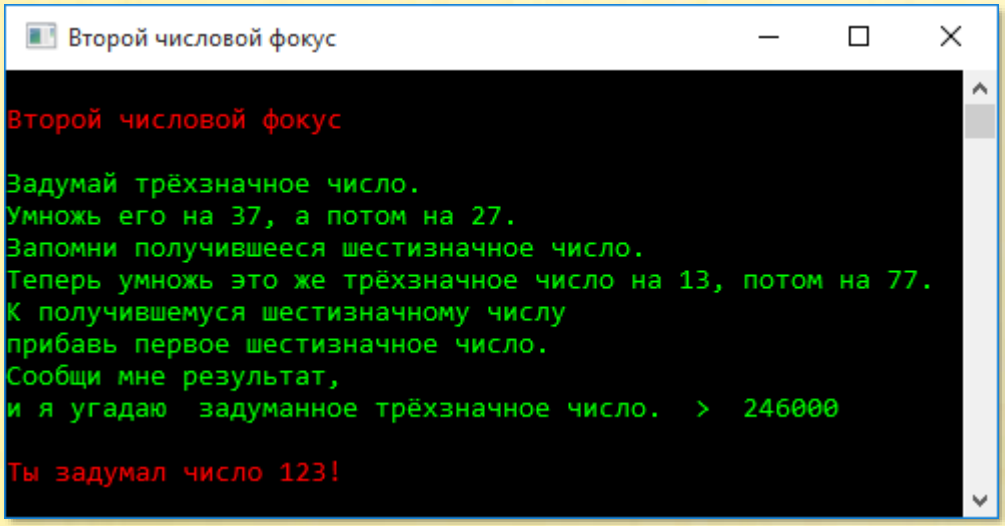
Всё сходится – фокус удался!

Вторую фокусную программу мы легко получим из первой. Главное - грамотно написать речь фокусника:

```
uses
   System;

// Кордемский, страница 35, задача 3

begin
   // заголовок окна:
   Console.Title := 'Второй числовой фокус';
   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Второй числовой фокус');

   // бесконечный цикл ввода данных -
   // пока пользователь не закроет программу
   // или не введет 0:
   repeat
      Console.ForegroundColor := ConsoleColor.Green;
      Console.WriteLine();

      Console.WriteLine('Задумай трёхзначное число.');
      Console.WriteLine('Умножь его на 37, а потом на 27.');
      Console.WriteLine('Запомни получившееся шестизначное число.');
      Console.WriteLine('Теперь умножь это же трёхзначное число на 13,
                         потом на 77.');
      Console.WriteLine('К получившемуся шестизначному числу');
      Console.WriteLine('прибавь первое шестизначное число.');
      Console.WriteLine('Сообщи мне результат,');
      Console.Write('и я угадаю  задуманное трёхзначное число.  >  ');
      var num := ReadInteger();
      if (num = 0) then exit;
```

```
      Solve(num);

      Console.ForegroundColor := ConsoleColor.Red;
      Console.WriteLine();
   until not (true);
end.


// ОТГАДЫВАЕМ ЧИСЛО
procedure Solve(num: integer);
begin
   Console.ForegroundColor := ConsoleColor.Red;
   if (num mod 2000 <> 0) then
   begin
        Console.WriteLine('Ты ошибся в вычислениях!');
        Console.WriteLine();
        exit;
   end;
   Console.WriteLine();
   Console.WriteLine('Ты задумал число {0}!', num div 2000);
   Console.WriteLine();
end;
```

Конечно, компьютер запросто разделит любое число на 2000, а вот загадчику придётся хорошенько потрудиться умножая и складывая числа!



Шестизначное число

Процедура без параметров
Цикл for
Оператор деления по модулю mod
Оператор continue
Оператор break

Задача 4 из книги *Удивительный мир чисел* [КА86], страница 44:

Ученику понадобилось написать наибольшее из шестизначных чисел, кратных 11 и чтобы цифра 6 была первой слева. Как надо действовать ученику для быстрого выполнения задания, если признаков делимости на 11 он ещё не знает?

Сообщим, что искомое число обладает забавной особенностью: если каждую его цифру повернуть на 180° в плоскости бумаги, оставляя её на прежнем месте, то образовавшееся число окажется дважды кратным 11 (делится на 11 и частное также делится на 11).

Выявите ещё одну особенность чисел: найденного и с повёрнутыми цифрами.

```
uses
   System;

// Кордемский, с.44, Задача 4

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.44, Задача 4';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Шестизначное число');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();
```

```
   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

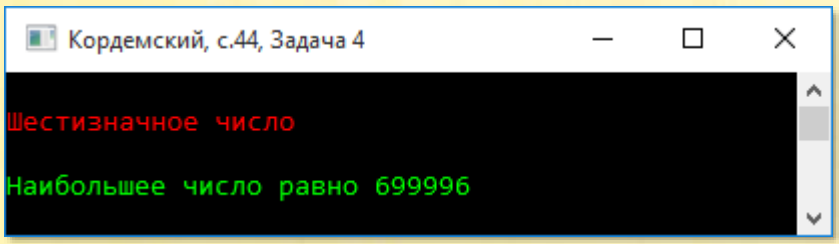
Легко догадаться, что нужно искать решение задачи среди шестизначных чисел, начинающихся с **шестёрки**:

```
//РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var min  := 600000;
   var max := 699999;
```

Причём перебирать числа следует задом наперёд, то есть от большего к меньшему, чтобы сразу же найти наибольшее из всех возможных чисел в заданном диапазоне:

```
   for var num := max downto min do
   begin
      // число должно быть кратно 11:
      if (num mod 11 <> 0) then
         continue;

      var s := 'Наибольшее число равно ' + num;
      Console.WriteLine(s);
      Console.WriteLine();
      break;
   end
end;
```

А вот и **ответ**:



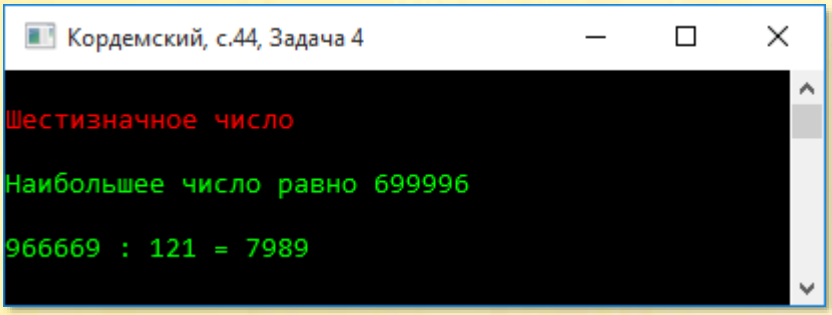
По условию задачи, **поворачиваем** все цифры найденного числа:

699996 → 966669

Проверяем, делится ли перевёрнутое число на 11 * 11 = 121:

```
if (966669 mod 121 = 0) then
    Console.WriteLine('966669 : 121 = ' + 966669 div 121)
else
    Console.WriteLine('Число 966669 на 121 не делится!');
```

Рисунок подтверждает: перевёрнутое число делится на 121:



А ещё одной особенностью этих чисел является их палиндромичность – они одинаково читаются и в прямом, и в обратном направлении.

Тройка, семёрка и... только

Процедура без параметров
Функция с параметром
Цикл for
Оператор continue
Условный оператор if
Тип данных int64
Оператор деления по модулю mod
Цикл while

Комбинированные операторы присваивания
Оператор ***exit***
Массив ***integer[]***

Задача 8 (13) из книги *Удивительный мир чисел* [КА86], страница 45:

> **Найдите наименьшее число**, обладающее следующими свойствами:
>
> - состоит только из цифр 7 и 3,
> - оно само и сумма его цифр делятся на 7 и на 3.

В отличие от *Пиковой дамы*, туза в этой задаче нет, но тройка и семёрка присутствуют в полном объёме!

```
uses
   System;

// Кордемский, с.45, Задача 8

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.45, Задача 8';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Тройка, семёрка и... только');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Предположим, что у нас имеется функция *Get37*, которая возвращает числа, составленные из цифр 3 и 7, и мы запоминаем в переменной min37 самое маленькое из них:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var min := 1;
   var max := 2400;
   // минимальное число:
   var min37 := int64.MaxValue;

   for var num := min to max do
   begin
      var ds := Get37(num);
      if (ds = -1) then continue;
```

Каждое такое число и сумма его цифр должны одновременно делиться на 3 и на 7, то есть на 21:

```
// число должно делиться на 3 и 7:
if (ds mod 21 <> 0) then continue;
// сумма цифр числа должна делиться на 3 и 7:
if (Summa(ds) mod 21 <> 0) then continue;
```

Если это так, то мы печатаем найденное число на экране и, если оно меньше текущего значения переменной *min37*, то запоминаем его:

```
      var s := 'Десятичное число равно ' + num;
      Console.WriteLine(s);
      s := 'Двоичное число равно ' + ds.ToString();
      Console.WriteLine(s);
      Console.WriteLine();

      // запоминаем наименьшее число:
      if (min37 > ds) then
         min37 := ds;
   end;
```

Просмотрев все числа в заданном диапазоне, мы печатаем ответ на задачу:

```
   Console.WriteLine('Наименьшее число равно ' + min37.ToString());
   Console.WriteLine();
end;
```

Функция Summa:

```
// НАХОДИМ СУММУ ЦИФР ЗАДАННОГО ЧИСЛА
function Summa(num: int64): integer;
begin
   var sum: int64 := 0;
   while (num > 0) do
   begin
      sum += num mod 10;
      num := num div 10;
   end;
   Result := integer(sum);
end;
```

Таким образом, нам осталось написать функцию Get37, которая умеет генерировать числа из троек и семёрок.

Поскольку во всех «подозреваемых» числах только 2 цифры, то вполне естественно представить их в *двоичной* системе. Путь **тройку** в двоичной записи будет символизировать **нуль**, а **семёрку** – **единица**, тогда несколько первых 3,7-чисел можно записать в двоичной системе так:

```
37 73 337 373 377 733 737 773 ...
01 10 001 010 011 100 101 110 ...
```

Между двоичными и 3,7-числами существует взаимно-однозначное соответствие (точно так же, как и между десятичными числами и их двоичным представлением), поэтому на вход функции *Get37* нужно подавать числа, начиная с нуля, и проверять в процедуре *Solve,* не является ли очередное 3,7-число решением за-

дачи. Но так как нас интересует не любое число, а именно **наименьшее**, а они генерируются не последовательно, то необходимо в процедуре *Solve* перебрать числа в некотором диапазоне, в котором заведомо окажется искомое число.

Перевести десятичное число в двоичное нетрудно, однако мы должны учесть, что нас интересует не оно, а соответствующее ему 3,7-число, поэтому каждый нуль в десятичном числе нужно заменить тройкой, а единицу – семёркой. Это легко сделать с помощью массива **idigs**, в который следует записать нужные цифры.

```
function Get37(num: integer): int64;
begin
   if (num < 1) then
   begin
      Result := -1;
      exit;
   end;

   // цифры числа:
   var idigs := new integer[](3, 7);
```

Также мы должны проследить, чтобы в числе обязательно присутствовали обе цифры – и тройка, и семёрка.

> Например, двоичным числам 00, 11, 000, 111 соответствуют 3,7-числа 33, 77, 333 и 777, которые целиком состоят из одинаковых цифр.

Для этого можно завести логический массив, подобный *idigs*, а можно просто объявить 2 логические **переменные**:

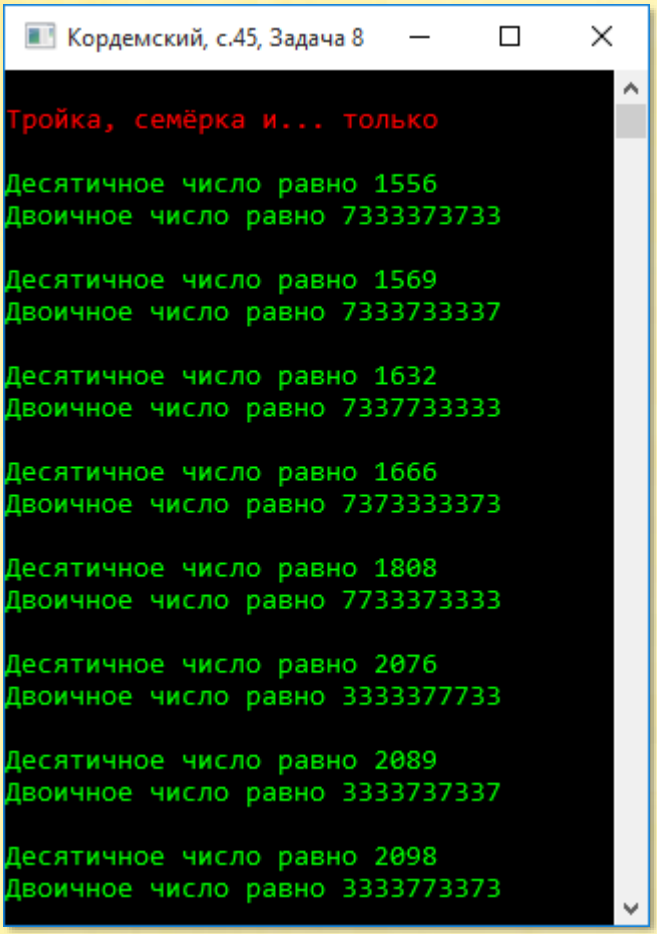
```
   // число из троек и семёрок:
   var num37: int64 := 0;
   // флажки для цифр 3 и 7:
   var flg3 := false;
   var flg7 := false;
```

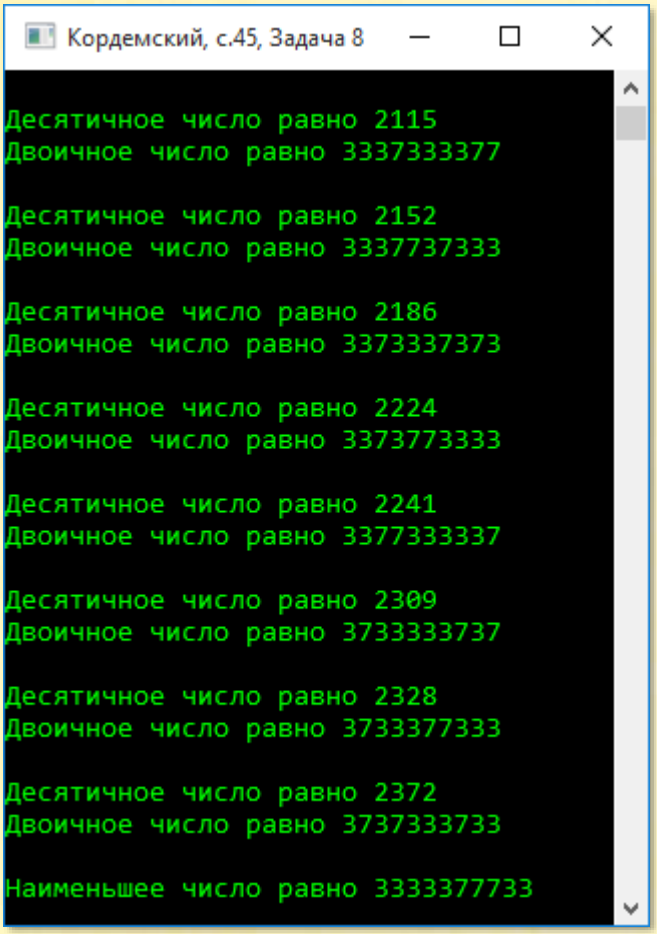
И последняя тонкость, которую необходимо «утолстить»: при конвертировании входного десятичного числа в двоичную форму мы никогда не получим ведущих нулей, поскольку число обратится в нуль, и деление на двойку закончится:

```
    // формируем число из троек и семёрок:
    var dec: int64 := 1;
    while (num > 2) do
    begin
       // очередная цифра:
       var dig := num mod 2;
       // отмечаем 3 и 7:
       flg3 := flg3 or (idigs[dig] = 3);
       flg7 := flg7 or (idigs[dig] = 7);
       num37 := num37 + idigs[dig] * dec;
       dec *= 10;
       num := num div 2;
    end;
    // строка должна включать и тройку, и семёрку:
    if (not flg3 or not flg7) then
    begin
       Result := -1;
       exit;
    end;

    Result := num37;
end;
```

Из этого следует, что нужно пожертвовать первой единицей в двоичном числе, тогда оставшиеся цифры дадут все комбинации нулей и единиц в двоичном числе и троек и семёрок – в 3,7-числе.

Решение задачи получилось не очень коротким, но зато программа нашла число за одно мгновение:





Как и в книге, это число – 3333377733.

Кроме этого, минимального 3,7-числа, существует бесконечно много 3,7-чисел, которые больше его.

> Можно формировать 3,7-число сзади, но тогда нужно отбросить последнюю единицу – см. функцию Get37R.

Любопытное свойство чисел

Процедура без параметров
Цикл ***for***
Оператор деления по модулю ***mod***
Оператор деления ***div***
Условный оператор ***if***
Оператор ***break***

Задача 15 (16 – номер задачи в издании 1996 года) из книги *Удивительный мир чисел* [КА86], страница 102:

> Возьмите какое-либо 6-значное число, делящееся на 7, например 325 836. Перенесите последнюю цифру в начало записи числа. Образуется новое число 632 583. Оно также делится на 7.
>
> **Докажите самостоятельно, что таким свойством обладает любое 6-значное число, делящееся на 7.**
>
> Рекомендация. Представьте заданное число так: 7k = = 10a+ 6 (1). Тогда новое число примет вид 100 0006 +a (2). Используя (1), докажите делимость (2) на число 7.

Доказать - значит, убедиться, что все 6-значные числа обладают указанным свойством. Это можно сделать «теоретически», то есть с помощью математических выкладок, а можно и «практически» - с помощью **метода грубой силы**, или **полного перебора**. Оба способа доказательства можно считать строгими, ведь они исчерпывают все возможные варианты. Поскольку 6-значных чисел совсем немного, то метод грубой силы здесь вполне уместен. Тем более что алгоритмы, построенные на этом методе почти всегда достаточно простые.

Традиционно из **главного блока** мы вызываем процедуру *Solve* для непосредственного решения задачи:

```
uses
   System;

// Кордемский, с.102, Задача 15

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.102, Задача 15';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Любопытное свойство чисел');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

В процедуре Solve мы обозначаем константами минимальное и максимальное 6-значные числа, которые нам и надлежит проверить:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
// мин. и макс. 6-значные числа:
const
   MIN6 = 100000;
   MAX6 = 999999;
begin
   var flg := true;
```

Флаг *flg* призван сигнализировать о (не)соблюдении условия задачи: поначалу он установлен в *true*, но если по ходу проверки нам встретится число, которое нарушает условие задачи, то мы с чистой совестью сбрасываем флаг и прекращаем дальнейшие проверки – гипотеза опровергнута. Если же после испытания всех 6-значных чисел флаг так и останется установленным в *true*, значит, гипотеза верна.

Поскольку диапазон значений проверяемых чисел точно очерчен, то перебирать числа удобнее всего в цикле *for*:

```
for var num := MIN6 to MAX6 do
   begin
```

Все числа, не кратные 7, мы пропускаем:

```
// число не кратно семи:
if (num mod 7 <> 0) then continue;
```

Всем остальным назначаем проверки.

Чтобы перенести последнюю цифру числа в начало, нужно её выделить. Это легко сделать, применив к числу операцию деления по модулю:

```
// последняя цифра числа:
var endn := num mod 10;
```

Число из первых 5 цифр найти ещё проще – нужно просто разделить исходное число на 10:

```
// 5-значное число из первых 5 цифр числа num:
var start5 := num div 10;
```

При формировании нового 6-значного числа мы переносим последнюю цифру в начало, что соответствует умножению на 100000. К этому произведению следует добавить 5-значное число из первых пяти цифр исходного числа:

```
// новое 6-значное число:
var newnum := endn * 100000 + start5;
```

Эту операцию можно записать более наглядно:

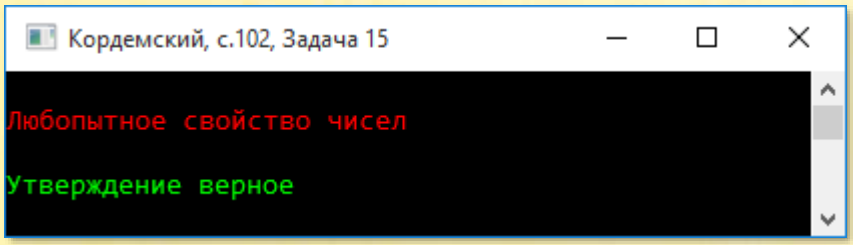
```
123456 ←исходное 6-значное число
612345 ←новое 6-значное число
```

Проверяем условие задачи:

```
      // если это число не кратно 7,
      // то утверждение неверное:
      if (newnum mod 7 <> 0) then
      begin
         flg := false;
         break;
      end;
   end;
```

Печатаем на экране результаты проверки:

```
   // все числа проверены:
   if (flg) then
        Console.WriteLine('Утверждение верное')
   else
        Console.WriteLine('Утверждение неверное');
   Console.WriteLine();
end;
```



Так как мы проверили все 6-значные числа, то можем уверенно утверждать, что любое 6-значное число удовлетворяет условиям задачи.

Как определил ошибку Чохбилмиш?

Процедура без параметров
Цикл ***for***
Оператор деления по модулю ***mod***
Оператор деления ***div***
Условный оператор ***if***
Оператор ***break***

Задача 23 (25) из книги *Удивительный мир чисел* [КА86], страница 57:

Чохбилмиш предложил каждому из двух учеников задумать какое-либо шестизначное число и переставить первую цифру в конец записи числа. Одному сказал: «Найди сумму получившихся чисел». Другому сказал: «Найди разность».

Ученики выполнили действия и написали:

913 485 и 167 860.

Чохбилмиш не знал, какие числа были задуманы учениками, но сразу определил: «Вы оба ошиблись».

Как рассуждал Чохбилмиш?

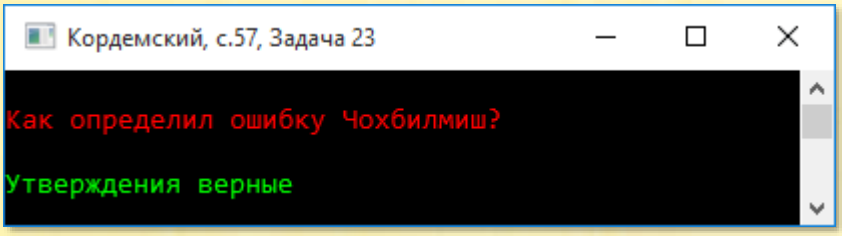
В ответе на задачу утверждается, что сумма любого 6-значного числа с другим 6-значным числом, полученным из исходного переносом первой цифры в конец числа, кратна 11 и аналогично полученная разность – кратна 9.

Мы можем проверить эти утверждения с помощью **полного перебора**, тем более что самая трудная часть задачи – перенос цифры – нами уже решена в предыдущем проекте.

```
uses
   System;

// Кордемский, с.57, Задача 23

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.57, Задача 23';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Как определил ошибку Чохбилмиш?');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
// мин. и макс. 6-значные числа:
const
   MIN6 = 100000;
   MAX6 = 999999;
begin
   var flg := true;

   for var num := MIN6 to MAX6 do
   begin
      // последняя цифра числа:
      var endn := num mod 10;
      // 5-значное число из первых 5 цифр числа num:
      var start5 := num div 10;
      // новое 6-значное число:
      var newnum := endn * 100000 + start5;
      // сумма и разность чисел num и newnum:
      var sum := num + newnum;
      var razn := num - newnum;
      // это число не кратно 11:
      if ((sum mod 11 <> 0) or (razn mod 9 <> 0)) then
      begin
         flg := false;
```

```
            break;
         end;
      end;
      // все числа проверены:
      if (flg) then
         Console.WriteLine('Утверждения верные')
      else
         Console.WriteLine('Утверждения неверные');
      Console.WriteLine();
end;
```

Проверив все 6-значные числа, мы можем с уверенностью констатировать факт: оба утверждения верные:



Всезнающая статистика

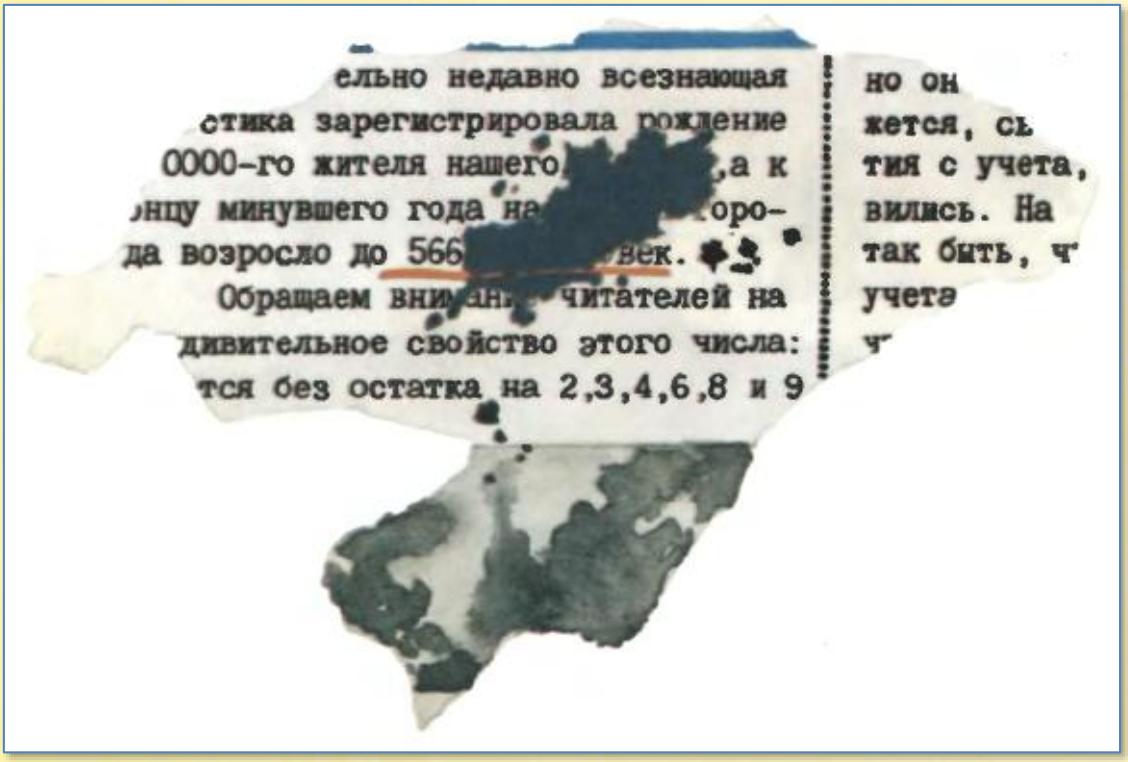
Процедура без параметров
Константы
Оператор ***continue***
Цикл ***for***
Оператор деления по модулю ***mod***

Задача 14 (16) из книги *Удивительный мир чисел* [КА86], страницы 86-87:

> Попал как-то мне в руки обрывок прошлогодней газеты. Моё внимание привлекло чернильное пятно, закрывшее последние три цифры шестизначного числа (Рис. 2.8). По сохранившемуся кусочку текста я вспомнил: это была заметка, в которой сообщалось, что к концу минувшего года

население нашего города возросло до этого числа. В заметке говорилось также о том, что это шестизначное число уникально среди шестизначных: оно делится на 2, 3, 4, 6, 7, 8 и 9. Ого! Не правда ли?

Чтобы восстановить все цифры этого числа, нет нужды обращаться в реставрационную лабораторию. Собственная сообразительность под-скажет вам математический метод быстрого решения этой задачи.



Опять всё решение задачи переносим в процедуру Solve:

```
uses
   System;

// Кордемский, с.86-87, Задача 14

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.86-87, Задача 14';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Всезнающая статистика');
```

```
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Эта задача очень простая, поскольку нам предстоит проверить всего одну тысячу чисел. Действительно, на обрывке газеты вы видите, что первые три цифры 6-значного числа – 566. Последние три – от 000 до 999:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
// мин. и макс. 6-значные числа:
const
   MIN6 = 566000;
   MAX6 = 566999;
```

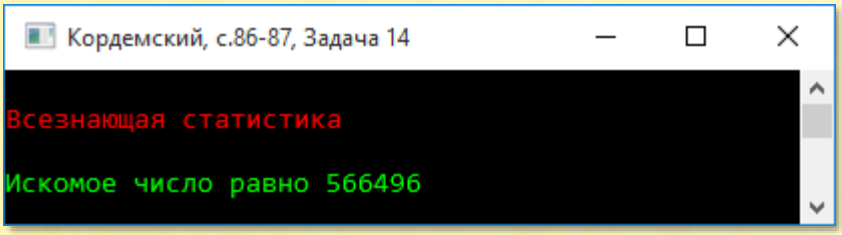
В цикле *for* мы перебираем все числа-кандидаты и проверяем их:

- Они должны нацело делиться на числа 2,3,4,6,7,8 и 9

```
begin
   for var num := MIN6 to MAX6 do
   begin
      // число не кратно 2,3,4,6,7,8,9:
      if (num mod 2 <> 0) then continue;
      if (num mod 3 <> 0) then continue;
      if (num mod 4 <> 0) then continue;
      if (num mod 6 <> 0) then continue;
      if (num mod 7 <> 0) then continue;
      if (num mod 8 <> 0) then continue;
      if (num mod 9 <> 0) then continue;

      // печатаем искомое число:
      Console.WriteLine('Искомое число равно ' + num);
```

```
    end;
    Console.WriteLine();
end;
```

Ответ вы можете видеть на рисунке:



Восстановите потерянную цифру

Процедура без параметров
Цикл ***for***
Оператор деления по модулю ***mod***

Задача 7 из книги *Удивительный мир чисел* [КА86], страница 85:

Числа вида $M_p = 2^p - 1$, где p - простое число, называются *числами Мерсенна*. При некоторых значениях *p* M_p - простое число. Так, первые одиннадцать простых чисел Мерсенна получаются при значениях *p* = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107.

Двенадцатое простое число Мерсенна равно $M_{127} = 2_{127} - 1$.

В его десятичной записи одна цифра стёрлась. Мы пока заменили её буквой *x*. Получилась такая запись:

170 141 183 460 469 231 *x*31 687 303 715 884 105 727.

Восстановите цифру, замещенную буквой *x*, если известно, что $M_{127} + 3$ делится на 13.

Задача решается очень просто, если вспомнить признак делимости чисел на 13:

```
uses
   System;

// Кордемский, с.85, Задача 7

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.85, Задача 7';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Восстановите потерянную цифру');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Цифра, обозначенная буквой ***x***, может принимать значения от 0 до 9. Их легко перебрать в цикле *for*

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   // арифметическая сумма чисел:
   var summa := 0;

   for var x := 0 to 9 do
   begin
```

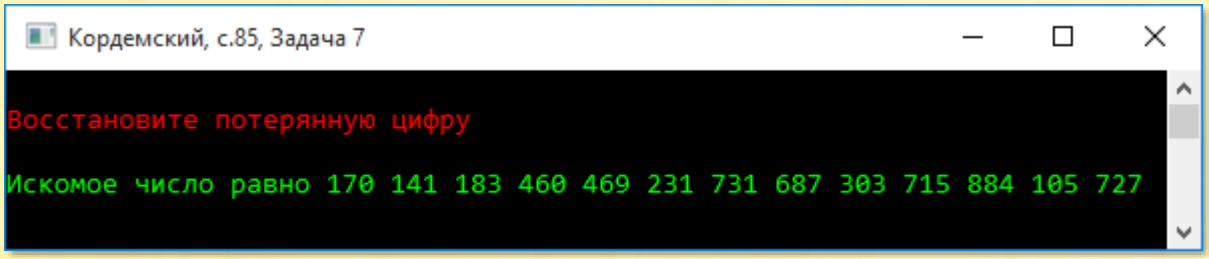
Для каждого значения *x* мы находим арифметическую сумму трёхзначных чисел – как и предписывает нам признак делимости на 13:

```
summa := 170 - 141 + 183 - 460 + 469 - 231 + x * 100 + 31 - 687 +
         303 - 715 + 884 - 105 + 727 + 3;
```

К получившейся сумме нужно добавить тройку. Если при этом переменная summa делится на 13, то задача решена:

```
      if (summa mod 13 <> 0) then continue;
      // печатаем искомое число:
      Console.WriteLine('Искомое число равно 170 141 183 460 469 231
                         {0}31 687 303 715 884 105 727', x);
   end;
   Console.WriteLine();
end;
```

На рисунке вы можете видеть, что x = 7:



Снимите маску с одной цифры

Процедура без параметров
Тип данных decimal
Цикл for
Оператор целочисленного деления mod
Условный оператор if

Задача 6 из книги *Удивительный мир чисел* [КА86], страница 84:

В записи знаменитого «шахматного» числа

M_{64}= 1y446 744 073 709 551 615

на его вторую цифру накинута маска *y*. Сняв маску, **расшифруйте значение *y***, зная, что достаточно увеличить заданное число на 3 единицы, как оно становится кратным числу 19.

Примечание. По легенде именно такое число пшеничных зёрен следовало выдать в награду изобретателю шахмат, попросившему положить всего одно зерно на первую клетку шахматной доски, а на каждую следующую клетку вдвое большее число зёрен, чем на предыдущую.

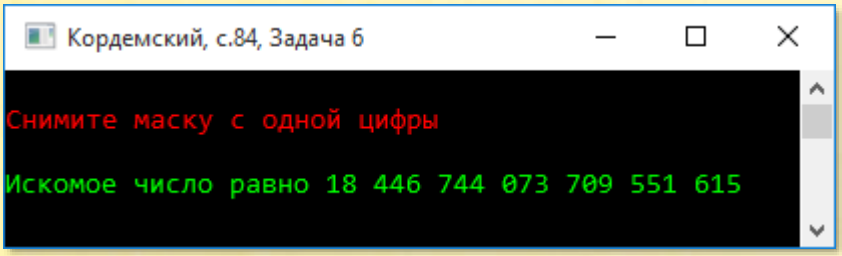
Эта задача сродни предыдущей, но признак делимости чисел на 19 не очень удобен:

```
uses
   System;

// Кордемский, с.84, Задача 6

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.84, Задача 6';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Снимите маску с одной цифры');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Поэтому в процедуре **Solve** мы просто находим остаток от деления числа *num + 3* на 19. Однако мы должны учесть, что искомое число очень велико, поэтому переменная *num* должна иметь тип **decimal**:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var num: decimal := 0;
   for var y := 0 to 9 do
   begin
      num := (10 + decimal(y)) * 1000000000000000000 +
                        446744073709551615 + 3;
      //Console.WriteLine('num = ' + num);
      if (num mod 19 <> 0) then continue;
      // печатаем искомое число:
      Console.WriteLine('Искомое число равно 1{0} 446 744 073 709
                         551 615', y);
   end;
   Console.WriteLine();
end;
```

Других сложностей в задаче нет, и мы быстро находим, что y = 8:



На одно делится, на другое нет

Процедура без параметров
Бесконечный цикл ***for***
Оператор целочисленного деления ***mod***
Оператор ***break***

Задача 2 из книги *Удивительный мир чисел* [КА86], страница 84:

Найдите наименьшее число, которое делится на 77, а при делении на 74 даёт в остатке 48.

Простая переборная задача – как раз для решения на компьютере:

```
uses
   System;

// Кордемский, с.84, Задача 2

begin
   //заголовок окна:
   Console.Title := 'Кордемский, с.84, Задача 2';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('На одно делится, на другое нет');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Поскольку искомое число кратно 77, то его можно представить в виде:

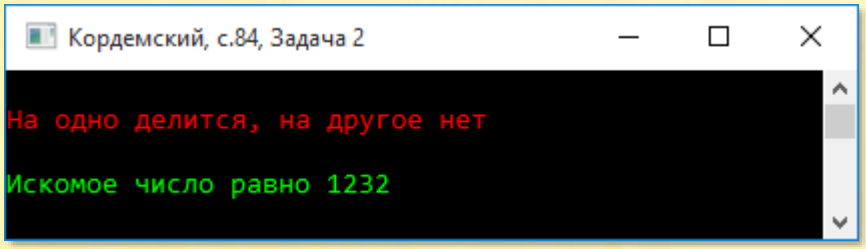
```
num = 77*n
```

Значение переменной n нужно изменять от 1 до ... - пока задача не будет решена. Здесь вполне уместно использовать *бесконечный цикл* ***for***, прерываемый оператором *break*, как только искомое число будет найдено:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
```

```
    for var n := 1 to integer.MaxValue do
    begin
      // искомое число кратно 77:
      var num := 77 * n;
      //Console.WriteLine('num = ' + num);
      if (num mod 74 <> 48) then continue;
      // печатаем искомое число:
      Console.WriteLine('Искомое число равно {0}', num);
      break;
    end;
    Console.WriteLine();
end;
```

Ответ на задачу показан на рисунке:



Кто где живёт?

Процедура без параметров
Массив integer[]
Массив string[]
Оператор цикла foreach
Цикл for
Цикл while
Условный оператор if-else
Оператор exit

Задача 3 из книги *Удивительный мир чисел* [КА86], страница 50:

Лайне, Майму, Юта, Белла, Елена и Элеонора живут в одном блоке здания. Возвращаясь из школы домой, Лайне проходит внутри здания 84 ступеньки, Майму -126, Юта -147, Белла - 189, Елена -210, а Элеонора -231 ступеньку. До квартиры на первом этаже ступенек нет, а между этажами одинаковые количества ступенек.

Кто на каком этаже живёт?

```
uses
   System;

// Кордемский, с.50, Задача 3

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.50, Задача 3';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Кто где живёт?');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Число ступенек, ведущих на заданный этаж можно представить в виде:

```
ЧИСЛО СТУПЕНЕК = ПРОЛЁТ*(ЭТАЖ-1)                    (1)
```

Откуда:

```
ЭТАЖ = (ЧИСЛО СТУПЕНЕК) / ПРОЛЁТ + 1                (2)
```

Единицу мы добавляем потому, что на первый этаж можно подняться без ступенек.

Число ступенек для каждого этажа нам известно из условия задачи. Эти данные, равно как и странные имена девочек, можно записать в **массив**, которым пользоваться удобнее, чем несколькими отдельными переменными:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   // число степенек:
   var stupeni := new integer[](84, 126, 147, 189, 210, 231 );
   // имена:
   var names := new string[]('Лайне',
                             'Майму', 'Юты',
                             'Беллы', 'Елены',
                             'Элеоноры');
```

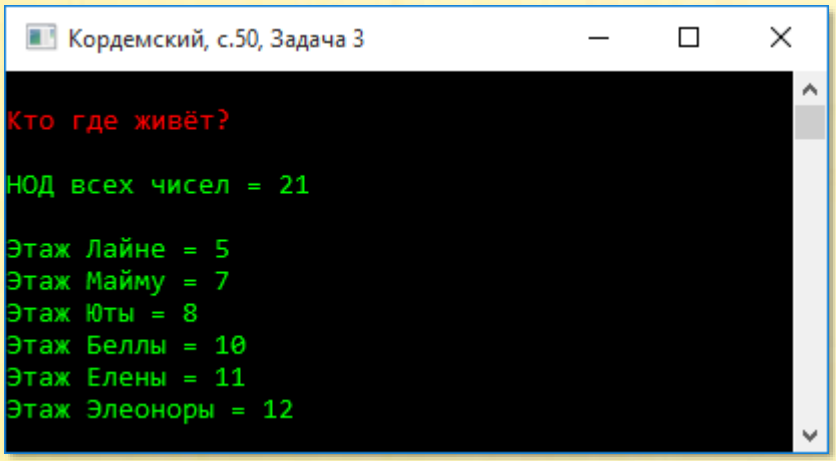
Число ступенек между всеми этажами (**ПРОЛЁТ**) одинаково во всём здании, а номер этажа выражается *целым* числом. Это значит, что число ступенек для каждого этажа должно делиться на одно и то же число **ПРОЛЁТ**. Все числа кратны их *наибольшему общему делителю*, который мы легко найдём с помощью функции **NOD**:

```
   // находим НОД всех чисел:
   var nodi := 84;
   foreach var i in stupeni do
   begin
      nodi := NOD(nodi, i);
   end;
   Console.WriteLine('НОД всех чисел = ' + nodi);
   Console.WriteLine();
```

Этажи, на которых находятся квартиры, теперь легко определить по формуле (2):

```
   // решение задачи:
   for var i := 0 to stupeni.Length-1 do
      Console.WriteLine('Этаж {0} = {1}', names[i],
                                 stupeni[i] div nodi + 1);

   Console.WriteLine();
end;


function NOD(n1, n2: integer): integer;
begin
   while (n2 <> n1) do
   begin
      if (n1 >= n2) then
         n1 -= n2
      else n2 -= n1;
   end;
   Result := n1;
end;
```

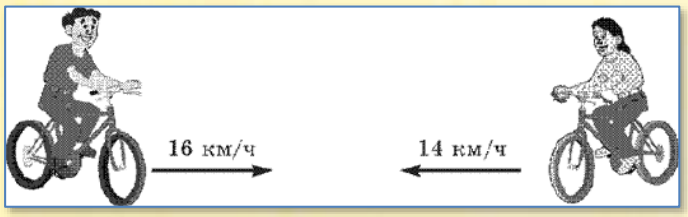
Ответ представлен на рисунке:



Так как 21 = 3 x 7, то число ступенек успешно можно разделить и на 3, и на 7. Номера этажей значительно увеличатся, но здравый смысл нам всё-таки подсказывает, что вряд ли между этажами всего 3 или 7 ступенек.

Три велосипедиста

Процедура без параметров
Функция с параметрами
Оператор exit
Условный оператор if
Цикл while
Оператор целочисленного деления mod
Бесконечный цикл repeat-until
Оператор and



Два вилисипидиста виихали навстричу друг другу из пунктов A и B…

Типичная задача по физике
и в исполнении нашего учителя

Задача 16 из книги *Удивительный мир чисел* [КА86], страница 55:

Три велосипедиста начали с общего старта движение по круговой дорожке. Первый делает полный круг за 21 мин, второй - за 35 мин, а третий - за 15 мин.

Через сколько минут они ещё раз окажутся вместе в начальном пункте?

```
uses
   System;

// Кордемский, с.55, Задача 16

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.55, Задача 16';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Три велосипедиста');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
```

```
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Задача решается элементарно, если догадаться, что велосипедисты окажутся в точке старта через **целое** число кругов. А это случится, когда пройдёт время *t* (в минутах), которое нацело делится на 21, 35 и 15. Из этого следует, что искомое время в точности равно **НОК** этих чисел:

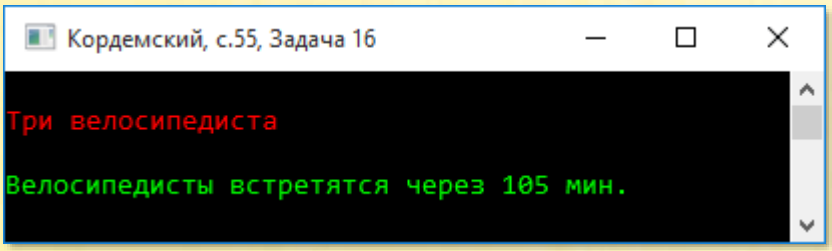
```
function NOD(n1, n2: integer): integer;
begin
   // если первое число меньше второго,
   // то меняем их значения:
   if (n1 < n2) then
   begin
      var n := n1;
      n1 := n2;
      n2 := n;
   end;
   // находим НОД:
   if (n2 = 0) then
   begin
      Result := n1;
      exit;
   end;
   while (n2 > 0) do
   begin
      var n := n1 mod n2;
      n1 := n2;
      n2 := n;
   end;
   Result := n1;
end;

function NOK(n1, n2: integer): integer;
begin
   Result := n1 * n2 div NOD(n1, n2);
end;

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   // находим НОК трёх чисел:
```

```
   var noki := NOK(NOK(21, 35), 15);

   Console.WriteLine('Велосипедисты встретятся через {0} мин.', noki);

   Console.WriteLine();
end;
```

Если вы хотите решить задачу короче, то нужно просто запустить часы и ждать, пока не пройдёт столько времени, чтобы оно делилось без остатка на означенные числа:

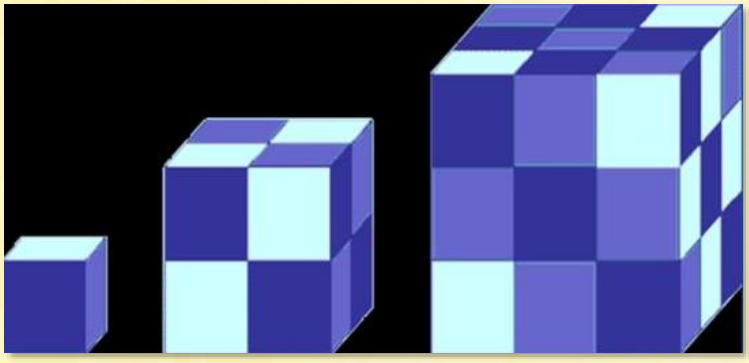
```
procedure Solve2();
begin
   var t := 0;
   repeat
      t += 1;
      if ((t mod 21 = 0) and (t mod 35 = 0) and (t mod 15 = 0)) then
         break;
   until not (true);

   Console.WriteLine('Велосипедисты встретятся через {0} мин.', t);
   Console.WriteLine();
end;
```

Но, как ни решай задачу, а встреча произойдёт через 105 минут:



Кубическое число

Процедура с параметром
Метод **Math.Round**
Метод **Math.Pow**
Цикл **for**
Тип данных **int64**
Условный оператор **if**
Оператор **break**



Задача 1 из книги *Удивительный мир чисел* [КА86], страница 89:

> **Найдите число**, куб которого - данное число:
>
> 1) **M** = 996 703 628 669;
>
> 2) **N** = 1 011 443 374 872.

Самый простой способ – извлечь из заданных чисел кубический корень.

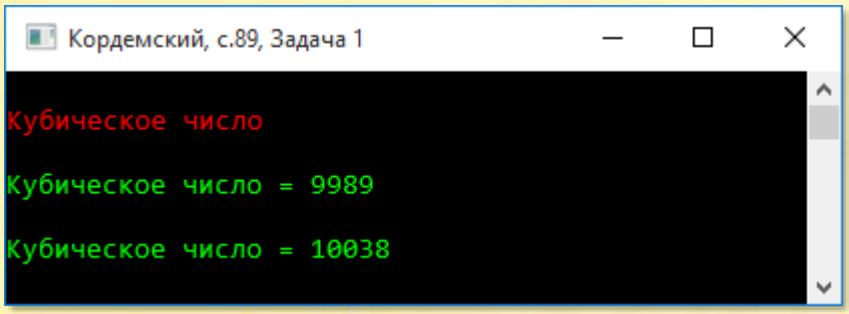
При этом мы должны учесть, что числа слишком велики для типа *integer*, но как раз впору типу **int64**. Это первое.

Второе: метод **Pow** не всегда возвращает точное значение, хотя по смыслу задачи оба корня должны быть целыми числами. В этом случае достаточно округлить корень до ближайшего целого числа методом *Round*:

```
uses
   System;

// Кордемский, с.89, Задача 1


// РЕШАЕМ ЗАДАЧУ
procedure Solve(num: int64);
begin
```

```
   Console.WriteLine('Кубическое число = ' +
                     Math.Round(Math.Pow(num, 1 / 3.0)));
   Console.WriteLine();
end;


begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.89, Задача 1';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Кубическое число');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve(996703628669);
   Solve(1011443374872);

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

На рисунке показан ответ на задачу:



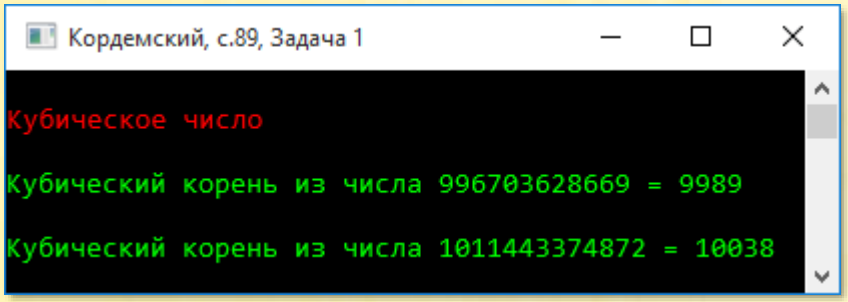
Поскольку кубические корни даже из очень больших чисел невелики, то можно найти их простым перебором, даже начиная с нуля!

Пишем вторую версию метода решения задачи:

```
procedure Solve2(num: int64);
begin
   var i: int64 := 0;
```

```
   while(true) do
begin
        if (i*i*i > num) then break;
        if (i*i*i = num) then
           Console.WriteLine('Кубический корень из числа {0} = {1}',
                             num, i);
        i += 1;
   end;
    Console.WriteLine();
end;
```

И получаем точные значения искомых корней:



При этом нам не понадобились методы класса *Math*!

Однако не забудьте в процедуре *Solve2* использовать переменную типа int64 - в противном случае вы получите бесконечный цикл – подумайте, почему!

И «хвост», и «грива»

Функция без параметров
Функция с параметром
Тип данных ***int64***
Оператор целочисленного деления ***mod***
Оператор ***exit***
Цикл ***while***
Вложенные циклы ***for***
Оператор ***or***

Задача 8 из книги *Удивительный мир чисел* [КА86], страница 72:

Если есть четырёхзначные числа, первые две и последние две цифры каждого из которых совпадают соответственно с первыми двумя и последними двумя цифрами квадрата и куба искомого числа, то какое из них наименьшее и какое - наибольшее? Числа, оканчивающиеся единицей и двумя нулями, исключаем.

В зашифрованном виде:

$xyzt^2 = xy...zt$ и

$xyzt^3 = xy...zt$.

В правой и левой частях этих равенств буквой ***x*** зашифрована одна и та же цифра, буквой ***y*** - также, буквой ***z*** - также и буквой ***t*** - также, но эти цифры могут быть и одинаковыми.

```
uses
   System;

// Кордемский, с.72, Задача 8

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.72, Задача 8';
```

```
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('И «хвост», и «грива»');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                     nVar.ToString());


   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Чтобы получить из квадратов и кубов две первые и две последние цифры (точнее, двузначные числа, составленные из этих цифр), мы напишем 2 вспомогательные функции.

Две последние цифры легко получить как остаток от деления исходного числа на сотню:

```
function Last2(num: int64): int64;
begin
   Result := num mod 100;
end;
```

С первыми двумя цифрами сложнее: мы не знаем «длину» числа, поэтому и не сможем сразу разделить его на нужную степень числа 10. В этом случае нужно последовательно делить заданное число на 10, пока оно не станет двузначным:

```
function First2(num: int64): int64;
begin
   while (num >= 100) do
   begin
      num := num div 10;
   end;
   Result := num;
```

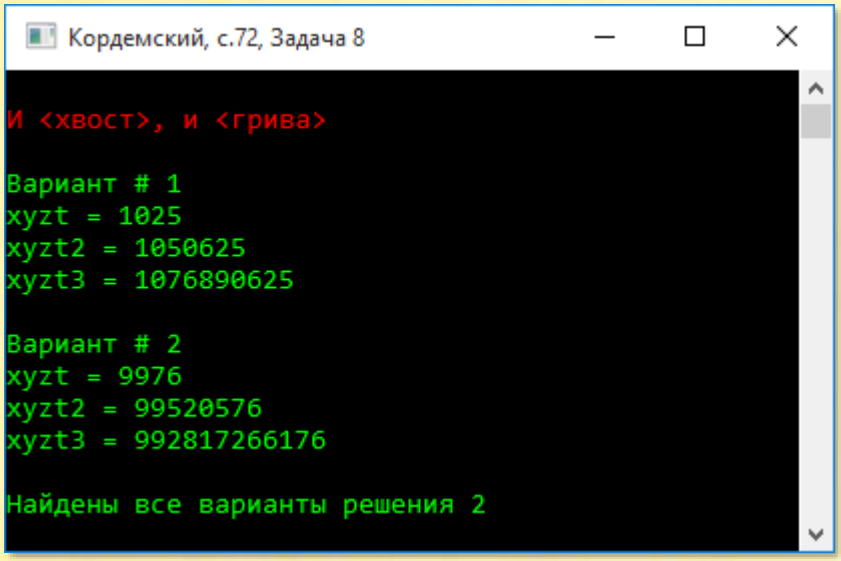
```
end;
```

Решить задачу можно так же, как мы решали числовые ребусы – с помощью четырёх вложенных циклов *for*. При этом мы учитываем, что переменные x, y, z, t могут иметь одинаковые значения, что в числовых ребусах недопустимо. Также из условия задачи следует, что *t не равно 0 и 1*, а *z не равно 0*.

В остальном функция Solve достаточна проста:

```
// РЕШАЕМ ЗАДАЧУ
function Solve() : integer;
begin
   Result := 0;

   for var x := 1 to 9 do
      for var y := 0 to 9 do
         for var z := 1 to 9 do
             for var t := 2 to 9 do
            begin
               var xyzt: int64 := x * 1000 + y * 100 + z * 10 + t;
               var xyzt2: int64 := xyzt * xyzt;
               var xyzt3: int64 := xyzt * xyzt * xyzt;

               if ((First2(xyzt2) <> x * 10 + y) or
                        (First2(xyzt3) <> x * 10 + y) or
                        (Last2(xyzt2) <> z * 10 + t) or
                        (Last2(xyzt3) <> z * 10 + t)) then
                  continue;

               Result += 1;
               Console.WriteLine('Вариант # ' +
                                 Result.ToString());
               Console.WriteLine('xyzt = ' + xyzt);
               Console.WriteLine('xyzt2 = ' + xyzt2);
               Console.WriteLine('xyzt3 = ' + xyzt3.ToString());
               Console.WriteLine();
            end
end;
```

Запустив программу, мы тут же получаем ответ:

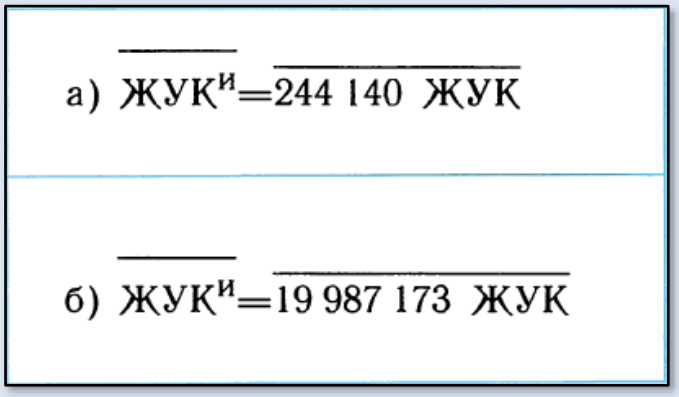


Так как функции First2 и Last2 позволяют извлекать нужные цифры из любых чисел, то задачу можно решить, проверяя в одном цикле *for* все четырёхзначные числа. Попробуйте!

Зашифрованные жуки

Функция без параметров
Вложенные циклы ***for***
Оператор ***continue***
Условный оператор ***if***
Оператор ***or***

Задача 4 из книги *Удивительный мир чисел* [КА86], страница 71:



Эту задачу вполне можно считать числовым ребусом и потому решать точно так же – с помощью вложенных циклов.

Все буквы, входящие в равенства а) и б), должны быть различными, а буква Ж не может равняться нулю.

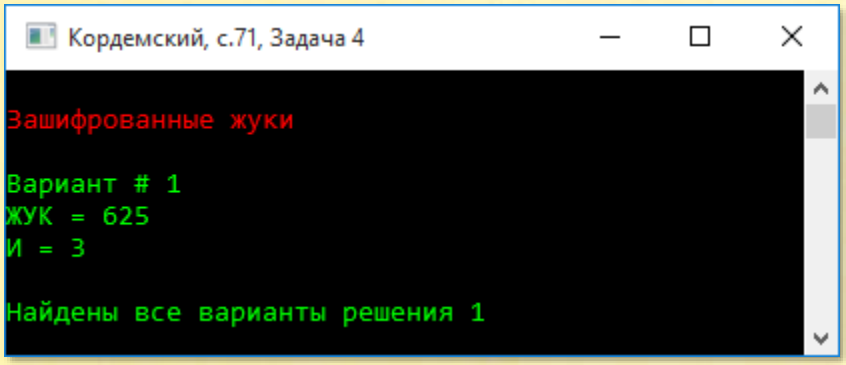
Значение степени И заведомо не меньше трёх и не больше шести.

```
uses
   System;

// Кордемский, с.71, Задача 4

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.71, Задача 4';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Зашифрованные жуки');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                     nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

С задачей **а**) мы справляемся без особых хлопот и затей:

```
// РЕШАЕМ ЗАДАЧУ
function Solve() : integer;
begin
   Result := 0;
   for var ZH := 1 to 9 do
      for var U := 0 to 9 do
      begin
         if (U = ZH) then continue;
         for var K := 0 to 9 do
         begin
            if ((K = U) or (K = ZH)) then continue;
            for var I := 3 to 6 do
            begin
               if ((I = K) or (I = U) or (I = ZH)) then continue;
               var ZHUK := ZH * 100 + U * 10 + K;
               var num: int64 := ZHUK;
               for var n := 2 to I do
                  num *= ZHUK;

               if (num <> 244140000 + ZH * 100 + U * 10 + K) then
                  continue;

               Result += 1;
               Console.WriteLine('Вариант # ' +
                                 Result.ToString());
               Console.WriteLine('ЖУК = ' + ZH + U + K);
               Console.WriteLine('И = ' + I);

               Console.WriteLine();
            end
         end
      end
end;
```

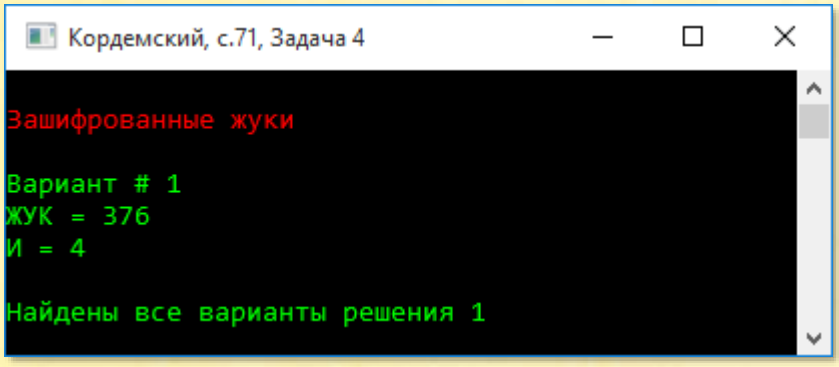
Она имеет *единственное* решение:



Для решения задачи б) нужно только изменить **проверку**:

```
//if (num <> 244140000 + ZH * 100 + U * 10 + K) then
//   continue;

if (num <> 19987173000 + ZH*100 + U*10 + K) then
   continue;
```

Она также имеет *единственное* решение:



Ж-Ж-Ж!

Функция без параметров
Цикл ***for***
Оператор ***exit***
Тип данных ***int64***
Условный оператор ***if***
Оператор ***continue***

Задача 3 из книги *Удивительный мир чисел* [КА86], страница 71:

$$(Ж-1)^5=\overline{Ж\ Ж\ Ж\ (Ж-1)}$$

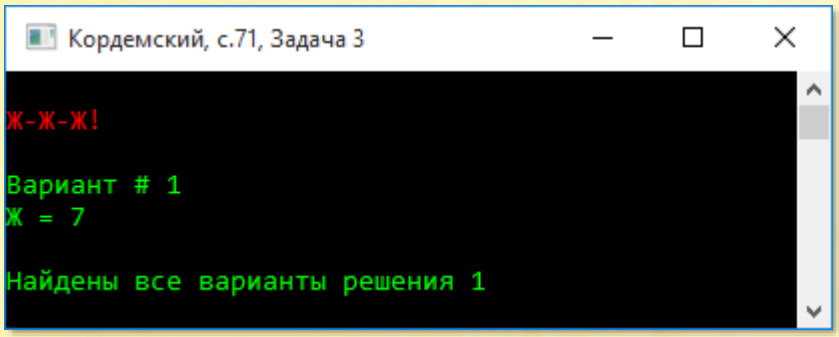
Убрав из жучиной программы всё лишнее, мы легко берём очередную задачу:

```
uses
   System;

// Кордемский, с.71, Задача 3


// РЕШАЕМ ЗАДАЧУ
function Solve() : integer;
begin
   Result := 0;
   for var ZH := 2 to 9 do
   begin
      var ZH1: int64 := ZH - 1;
      var num: int64 := ZH1;
      for var i := 2 to 5 do
         num *= ZH1;

      if (num <> ZH * 1000 + ZH * 100 + ZH * 10 + ZH1) then
         continue;

      Result += 1;
      Console.WriteLine('Вариант # ' +
                        result.ToString());
      Console.WriteLine('Ж = ' + ZH);

      Console.WriteLine();
   end
end;


begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.71, Задача 3';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Ж-Ж-Ж!');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
```

```
                        nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

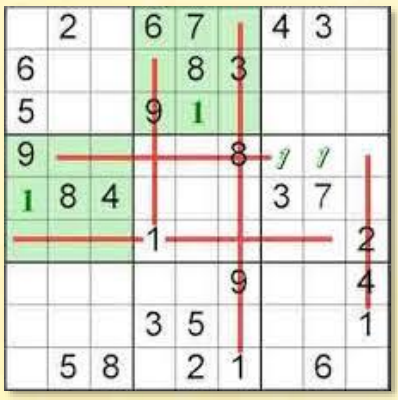
На рисунке вы видите, что Ж = 7, то есть полное решение такое:

$(7-1)^5 = 7776$



Девять в квадрате

Функция без параметров
Цикл ***for***
Тип данных ***int64***
Оператор целочисленного деления ***mod***
Оператор ***exit***



Задача 1 из книги *Удивительный мир чисел* [КА86], страница 70:

Найдите шестизначное число, зашифрованное в ребусе:

ДЕВЯТЬ^2=§§§§§§ДЕВЯТЬ

Мы легко найдём, что **наименьшее** 6-значное число, состоящее из разных цифр, равно 102345, а **наибольшее** – 987654. Таким образом, нужно перебрать все числа в этом диапазоне, возвести их в квадрат и сравнить последние 6 цифр квадрата с исходным числом. По условию задачи, они должны совпадать.

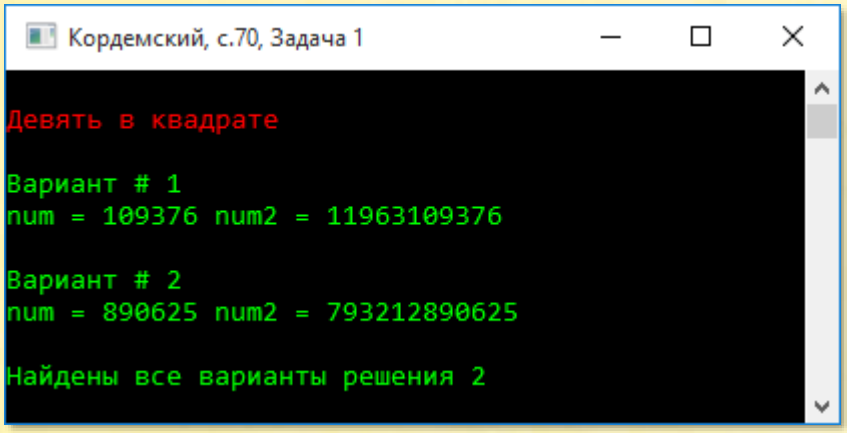
```
uses
   System;

// Кордемский, с.70, Задача 1

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.70, Задача 1';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Девять в квадрате');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                        nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

В функции Solve мы и выполняем означенные действия:

```
// РЕШАЕМ ЗАДАЧУ
function Solve() : integer;
begin
   Result := 0;
   var min: int64 := 102345;
   var max: int64 := 987654;
   for var num: int64 := min to max do
   begin
      var num2: int64 := num * num;
      if (num2 mod 1000000 <> num) then
         continue;
```

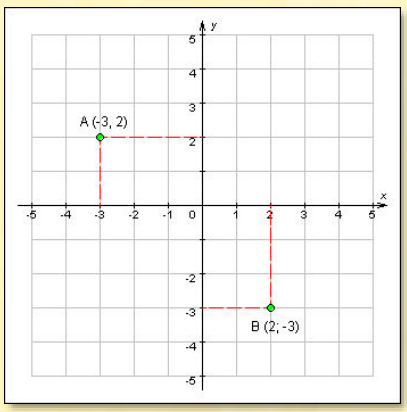
```
      Result += 1;
      Console.WriteLine('Вариант # ' +
                                Result.ToString());
      var s := 'num = ' + num.ToString;
      s += ' num2 = ' + (num * num).ToString();
      Console.WriteLine(s);
      Console.WriteLine();
   end
end;
```

Задача имеет 2 решения:



Пара чисел: 3149 и 3151

Функция без параметров
Цикл for
Тип данных int64
Условный оператор if
Оператор continue



Задача 3.2 из книги *Удивительный мир чисел* [КА86], страница 43:

Десятичная запись куба каждого из чисел 3149 и 3151 начинается двумя его первыми цифрами, а оканчивается двумя его последними цифрами:

3149^3 = 31 226 116 949
3151^3 = 31 285 651 951

Есть ещё два последовательных четырёхзначных числа, обладающих таким же свойством.

Какие это числа?

Задача решается полным перебором всех четырёхзначных чисел:

```
uses
   System;

// Кордемский, с.43, Задача 3-2

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.43, Задача 3-2';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Пара чисел: 3149 и 3151');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                     nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.


// РЕШАЕМ ЗАДАЧУ
function Solve() : integer;
begin
   Result := 0;
   var min := 1000;
```

```
    var max := 9999;

    for var num: int64 := min to max do
    begin
```

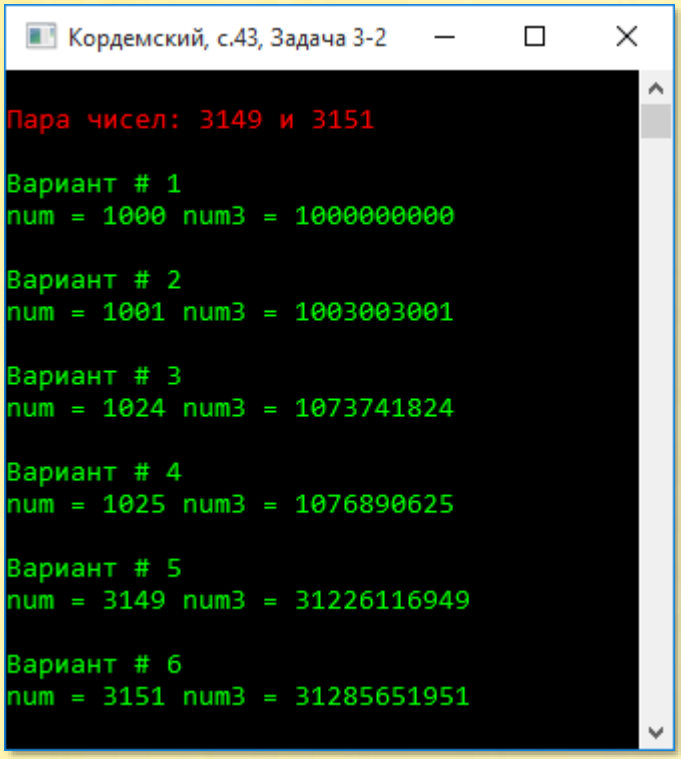
Для каждого четырёхзначного числа мы находим его куб.

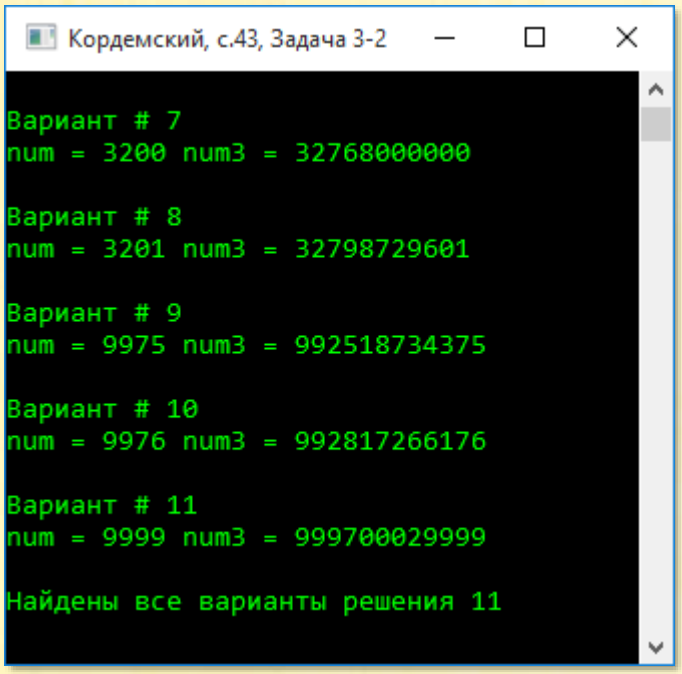
> Числа большие, поэтому они должны быть типа **int64**!

Затем сравниваем в этих числах 2 первые и 2 последние цифры. Для этого мы привлекаем функции First2 и Last2, разработанные нами в проекте *И «хвост», и «грива»*:

```
        var num3: int64 := num * num * num;
        // две последние цифры:
        if (Last2(num) <> Last2(num3)) then
           continue;
        // две первые цифры:
        if (First2(num) <> First2(num3)) then
           continue;

        Result += 1;
        Console.WriteLine('Вариант # ' +
                                Result.ToString());
        var s := 'num = ' + num;
        s += ' num3 = ' + (num * num * num).ToString();
        Console.WriteLine(s);
        Console.WriteLine();
    end
end;
```

На рисунке приведён полный список чисел, выполняющих условие задачи.





В книге указана пара чисел 3200 и 3201, но вы видите, что есть и другие пары чисел:

1000 и 1001
1024 и 1025
9975 и 9976

Число 698 896

Функция без параметров
Тип данных ***int64***
Цикл ***for***
Функция *с параметром*
Цикл ***while***
Оператор ***exit***
Метод ***Math.Max***
Оператор целочисленного деления ***mod***

Задача 3.3 из книги *Удивительный мир чисел* [КА86], страница 43:

> Число 698 896 *квадратное* (836^2), *палиндромическое*. Предполагают, что оно - наименьшее квадратное палиндромическое число с чётным количеством цифр. Подтвердить это или опровергнуть можно, только покопавшись в таблице квадратов чисел, меньших 836.

И правда, давайте покопаемся!

```
uses
   System;

// Кордемский, с.43, Задача 3-3

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.43, Задача 3-3';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Число 698 896');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                        nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Например, можно докопаться до чисел, гораздо больших, чем 836:

```
// РЕШАЕМ ЗАДАЧУ
function Solve() : integer;
begin
   Result := 0;
   var min := 4;
```

```
   var max := 1000000;

   for var num: int64 := min to max do
   begin
      var num2: int64 := num * num;
```

Чтобы распознать квадратный палиндром, мы пишем функцию IsPalindromic-Number:

```
function IsPalindromicNumber(num: int64): boolean;
begin
   // избавляемся от чисел, заканчивающихся на нуль:
   if ((num <> 0) and (num mod 10 = 0)) then
   begin
      Result := false;
      exit;
   end;

   var rev: int64 := 0;
   while (num >= rev) do
   begin
      rev := 10 * rev + num mod 10;
      if (num = rev) then
      begin
         Result := true;
         exit;
      end;

      num := num div 10;
      if (num = rev) then
      begin
         Result := true;
         exit;
      end;
   end;
   Result := false;
end;


// если не палиндром:
if (!IsPalindromicNumber(num2))
   continue;
```

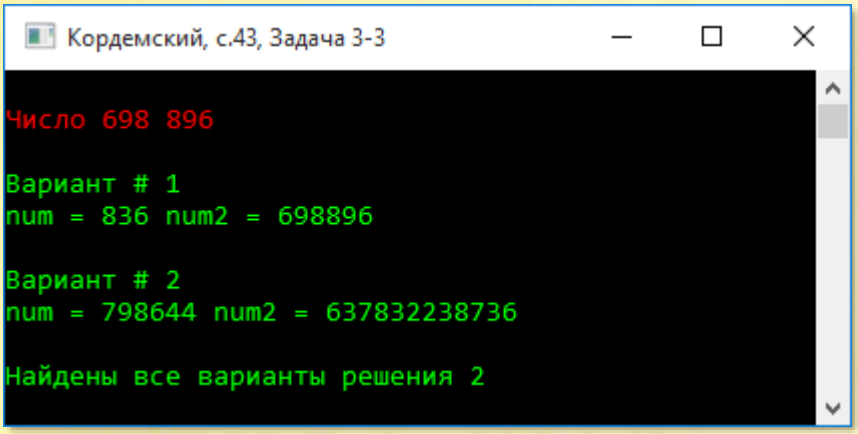
А для подсчёта цифр в квадрате нам также необходимо написать новую функцию:

```
// НАХОДИМ ЧИСЛО ЦИФР В ЗАДАННОМ ЧИСЛЕ
function NumDigit(num: int64): integer;
begin
   num := Math.Abs(num);
   var nd := 0;
   while (num > 0) do
   begin
      nd += 1;
      num := num div 10;
   end;
   Result := Math.Max(1, nd);
end;
```

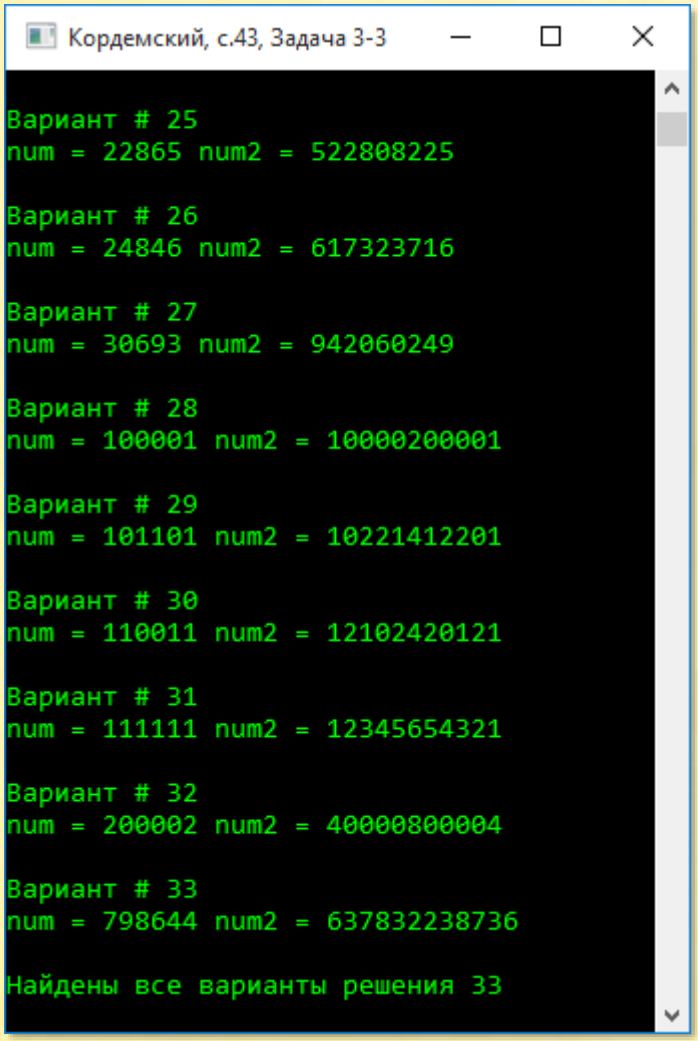
Как и полагается по условию задачи, мы рассматриваем только числа с чётным набором цифр:

```
      // если не палиндром:
      if (not IsPalindromicNumber(num2)) then
         continue;

      // число цифр:
      if (NumDigit(num2) mod 2 <> 0) then
         continue;

      Result += 1;
      Console.WriteLine('Вариант # ' +
                        Result.ToString());
      var s := 'num = ' + num.ToString();
      s += ' num2 = ' + (num * num).ToString();

      Console.WriteLine(s);
      Console.WriteLine();
   end
end;
```

За полсекунды мы, покопавшись и порывшись, установили, что первое квадратное палиндромическое число именно 698 896, а второе – гораздо больше:



Если искать квадратные палиндромы с **любым** числом цифр, то можно откопать и накопать весьма любопытные экземпляры:



Числа 11 826, 12 363, 14 676

Функция без параметров
Цикл ***for***
Тип данных ***int64***
Условный оператор ***if***
Оператор ***continue***
Оператор ***exit***
Функция с параметрами
Массив ***integer[]***
Цикл ***while***
Метод ***Array.IndexOf***
Оператор ***or***

Задача 3.4 из книги *Удивительный мир чисел* [КА86], страница 43:

Числа 11 826, 12 363, 14 676. В десятичной записи квадрата каждого из них содержатся цифры от 1 до 9, по одному разу каждая:

11 8262= 139 854 276
12 3632 = 152 843 769
14 6762 = 215 384 976

Есть ещё пятизначное число с таким же свойством. Его запись содержит цифры 1, 2, 3, 4, 5, каждую по одному разу. Из пяти разных цифр (без нуля) можно сформировать 5! = 120 разных пятизначных чисел (убедитесь!), но, пользуясь калькулятором, вы без большого труда выделите из них требуемое число.

Компьютер именно без труда большого труда переберёт все пятизначные числа:

```
uses
   System;

// Кордемский, с.43, Задача 3-4
```

```
begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.43, Задача 3-4';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Числа 11 826, 12 363, 14 676');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                     nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.


// РЕШАЕМ ЗАДАЧУ
function Solve() : integer;
begin
   Result := 0;
   var min := 10000;
   var max := 99999;

   for var num: int64 := min to max do
   begin
      var num2: int64 := num * num;
```

Единственная проблема – выяснить, что цифры в числе не повторяются и что среди них нет нуля. Нам ничего не остаётся делать, как написать новую функцию для этого:

```
      // если цифры повторяются:
      if (not IsNoRepDigit(num2)) then
      //if (not IsNoRepDigit(num2, false)) then
         continue;

      Result += 1;
      Console.WriteLine('Вариант # ' +
                        result.ToString());
      var s := 'num = ' + num.ToString();
```

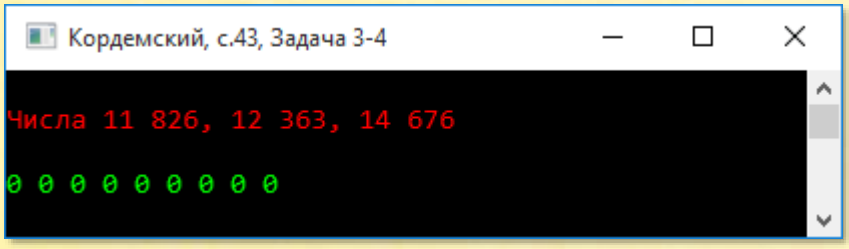
```
      s += ' num2 = ' + num2.ToString();

      Console.WriteLine(s);
      Console.WriteLine();
   end
end;
```

Можно придумать много способов, как проверить, что в числе цифры не повторяются. Самый *простой* из них – конвертировать число в строку и проверять символы. Но *правильнее* сделать так.

С помощью функции *NumDigit* мы узнаем, сколько всего цифр в числе и создадим для них массив digs по числу цифр:

```
// ОПРЕДЕЛЯЕМ, ЧТО ВСЕ ЦИФРЫ В ЗАДАННОМ ЧИСЛЕ
// РАЗНЫЕ
function IsNoRepDigit(num: int64; noNull: boolean := true): boolean;
begin
   // число цифр в числе:
   var nd := NumDigit(num);
   // помещаем их в массив:
   var digs := new integer[nd];
```

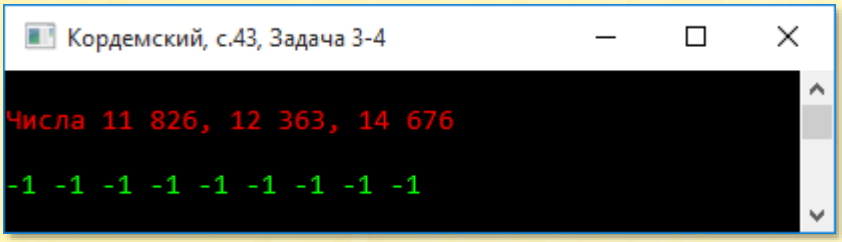
Сразу после создания в массиве все элементы будут иметь значение 0:



Если в заданном числе num также имеется нуль, то такое число не пройдёт проверки, даже если оно состоит из разных цифр. Поведение функции *IsNoRepDigit* мы регулируем с помощью параметра noNull. По умолчанию он равен *true*, то есть числа, содержащие нуль, будут отвергаться.

Если же мы допускаем в числе присутствие нуля, то значение параметра noNull должно быть *false*. В этом случае мы все элементы массива *digs* обращаем в -1, которой заведомо нет среди цифр заданного числа:

```
// в числе нет нуля:
if (not noNull) then
   for var i := 0 to nd - 1 do
      digs[i] := -1;
```



Если заданное число равно нулю, то функция *IsNoRepDigit* возвращает *true* или *false* в зависимости от того, допускается ли в заданном числе нуль:

```
if ((num = 0) and (not noNull)) then
begin
   Result := true;
   exit;
end;
if ((num = 0) and noNull) then
begin
   Result := false;
   exit;
end;

num := Math.Abs(num);
var n := 0;
```

Если число отличается от нуля, то мы начинаем извлекать из него по одной цифре, начиная с конца:

```
while (num > 0) do
```

```
    begin
      var dig := (integer)(num mod 10);
```

И вот здесь мы должны убедиться, что такая цифра ещё не встречалась, то есть её нет в массиве *digs*. Для этого мы вызываем метод IndexOf класса *Array*, который возвращает индекс в массиве *digs* с элементом заданного значения *dig*. Если же такой элемент в массиве отсутствует, то этот метод вернёт -1. Это значит, что такую цифру мы уже ранее поместили в массив *digs*, и цифра *dig* повторяется в заданном числе:

```
      if (System.Array.IndexOf(digs, dig) <> -1) then
      begin
        Result := false;
        exit;
      end;
      digs[n] := dig;
      n += 1;
      num := num div 10;
    end;
    //digs.Println();
    Result := true;
end;
```

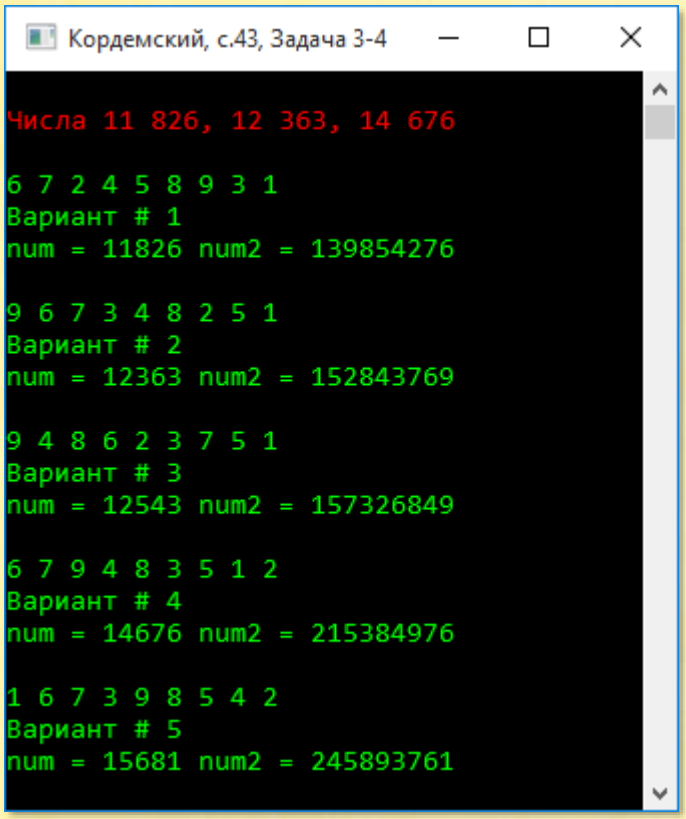
Если вы раскомментируете строчку

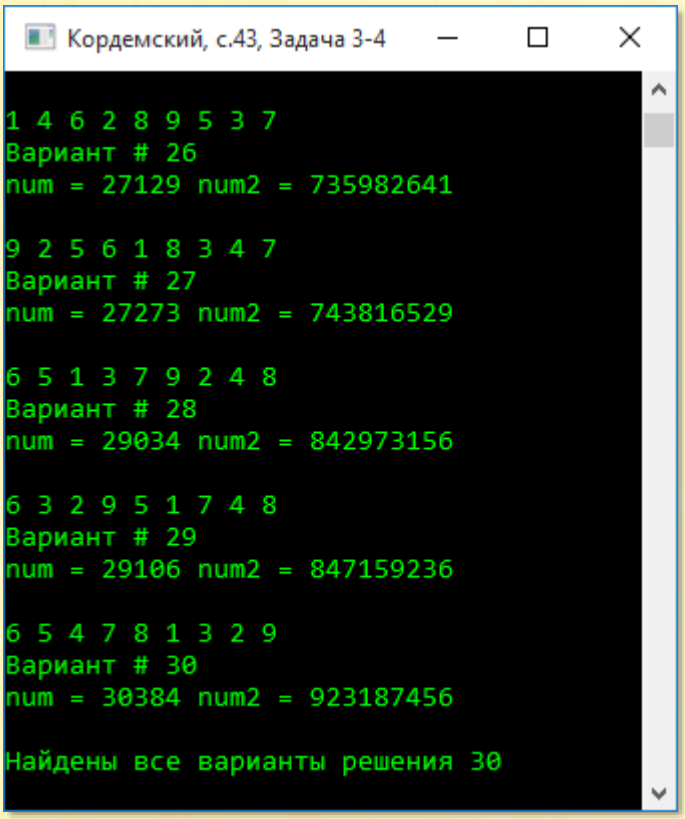
```
//digs.Println();
```

то увидите в Консольном окне распечатку массива digs для всех чисел с неповторяющимися цифрами.

На этом подготовительные работы заканчиваются, и мы приступаем к решению задачи.

На рисунке вы можете видеть часть списка пятизначных чисел, квадраты которых состоят из разных цифр и не содержат нуля. Всего таких чисел ровно 30:



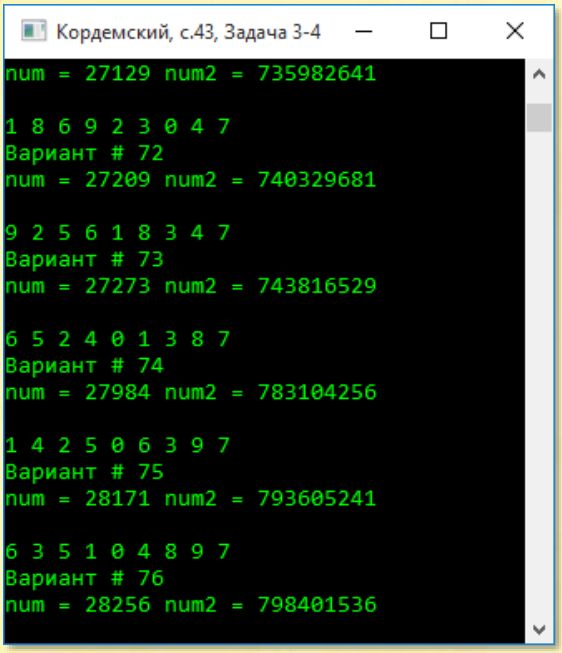


Что касается числа, состоящего из цифр 1,2,3,4,5, то оно в списке занимает третье место. Это число **12543**.

Если вы пожелаете найти разноциферные числа с **нулём**, то перенесите комментарий на строчку выше:

```
// если цифры повторяются:
//if (not IsNoRepDigit(num2)) then
if (not IsNoRepDigit(num2, false)) then
```

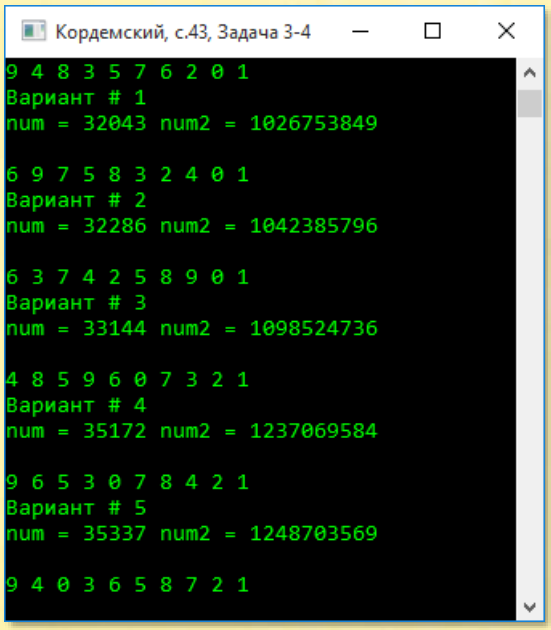
Как и следовало ожидать, чисел с неповторяющимися цифрами стало заметно больше, но в некоторых из этих чисел отсутствует, например, цифра 8, так что ряд цифр получается разорванным:

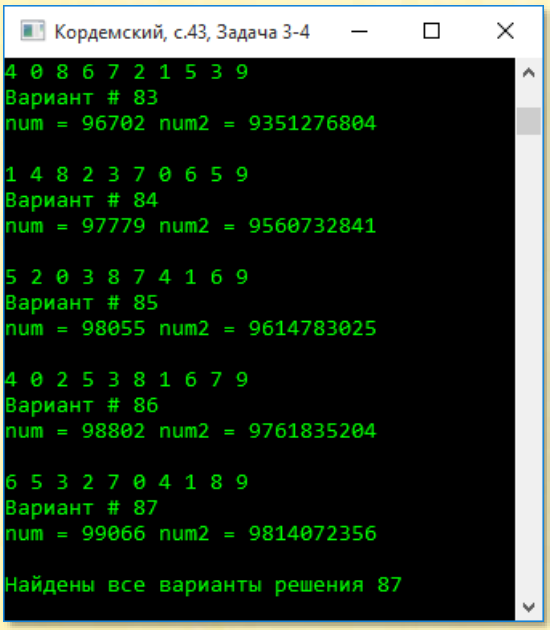


Тут, пожалуй, лучше изменить условие проверки чисел так, чтобы были напечатаны только 10-значные числа:

```
if ((not IsNoRepDigit(num2, false)) or (num2 < 1000000000)) then
   continue;
```

Их оказалось на удивление много:





Числа 32 043 и 99 066

Задача 3.8 из книги *Удивительный мир чисел* [КА86], страница 44:

> **Числа 32 043 и 99 066.** В запись квадрата каждого из них входят все 10 цифр, по одному разу каждая:
>
> $32\,043^2 = 1\,026\,753\,849$
> $99\,066^2 = 9\,814\,072\,356.$
>
> Предполагают, что первое - наименьшее, а второе - наибольшее из пятизначных чисел с таким свойством.
>
> Подтвердить или опровергнуть это утверждение под силу только ЭВМ или энтузиасту - любителю счёта.

Как верно отмечено в условии задачи, это нам под силу! Более того, мы ненароком уже решили эту проблему в предыдущем проекте, и на последнем рисунке отчётливо видно, что указанные числа действительно наименьшее и наибольшее из всех пятизначных чисел.

Число 117 649

Функция без параметров
Тип данных **int64**
Цикл **for**
Условный оператор **if**
Оператор **continue**
Метод **Math.Round**
Функция с параметром

Задача 3.6 из книги *Удивительный мир чисел* [КА86], страница 44:

> **Число 117 649** существует одновременно в трёх качествах. Оно:
>
> - *квадратное*
> - *кубическое*
> - *кратное семи*:
>
> $117\,649 = 343^2 = 49^3 = 7k,\ k \in N.$
>
> Более того, на отрезке от единицы до миллиона оно единственное с таким свойством.
>
> **Докажите!**

Чтобы доказать это утверждение, нужно проверить все числа в заданном диапазоне (но мы расширим диапазон, чтобы найти и другие числа).

```
uses
   System;

// Кордемский, с.43, Задача 3-6


begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.43, Задача 3-6';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Число 117 649');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                        nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   Result := 0;
   var min := 7;
   var max := 1000000000;
```

Чтобы не проверять делимость чисел на 7, можно начать поиск с семёрки и в каждой следующей итерации добавлять к переменной цикла также семёрку, тогда все проверяемые числа будут сразу кратны семи, и одна проверка отпадёт сама собой.

```
for var n: int64 := min div 7 to max div 7 do
   begin
      var num:= n*7;
      // число должно быть полным квадратом и кубом:
```

Проверку числа на *квадратность* и кубичность мы доверим двум функциям:

```
 if (not IsQuadrat(num)) then
    continue;
 if (not IsQube(num))    then
    continue;
```

Если очередное число num выдержало все проверки, то оно является решением задачи, и мы печатаем его на экране:

```
      Result += 1;
      Console.WriteLine('Вариант # ' +
                        Result.ToString());
      var s := 'num = ' + num.ToString();
      Console.WriteLine(s);
      s := 'Квадратный корень = ' +
           Math.Round(Math.Pow(num, 1.0 / 2.0));
```

```
      Console.WriteLine(s);
      s := 'Кубический корень = ' +
           Math.Round(Math.Pow(num, 1.0 / 3.0));
      Console.WriteLine(s);
      Console.WriteLine();
   end
end;
```

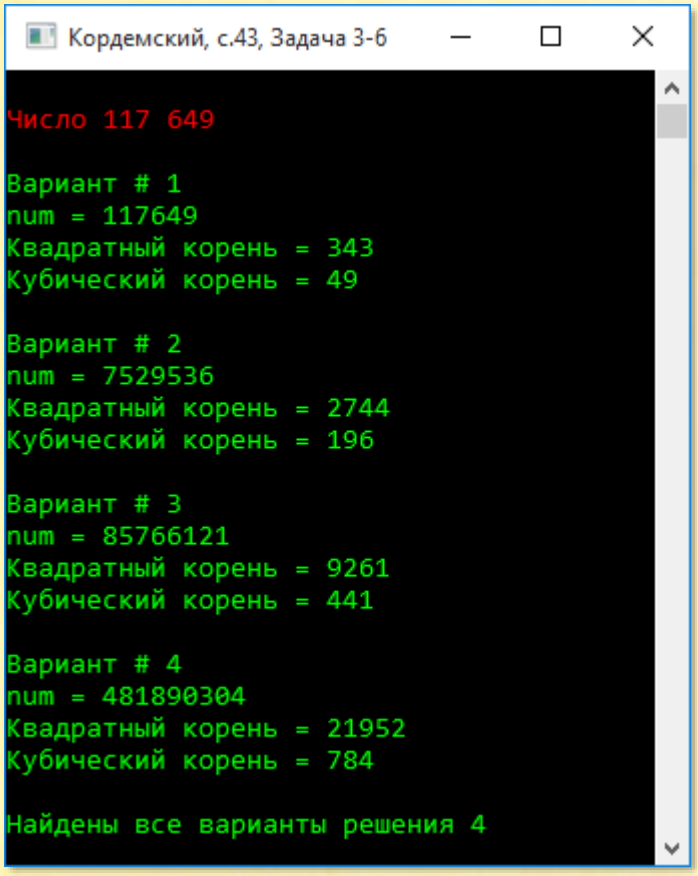
Проверку числа на *квадратность* можно выполнить с помощью методов Sqrt или Pow класса *Math*:

```
// ОПРЕДЕЛЯЕМ ЯВЛЯЕТСЯ ЛИ ЗАДАННОЕ ЧИСЛО
// ПОЛНЫМ КВАДРАТОМ
function IsQuadrat(num: int64): boolean;
begin
   var r := Trunc(Math.Sqrt(num));
   Result := r * r = num;
end;
```

А на *кубичность* с помощью метода Pow класса *Math*:

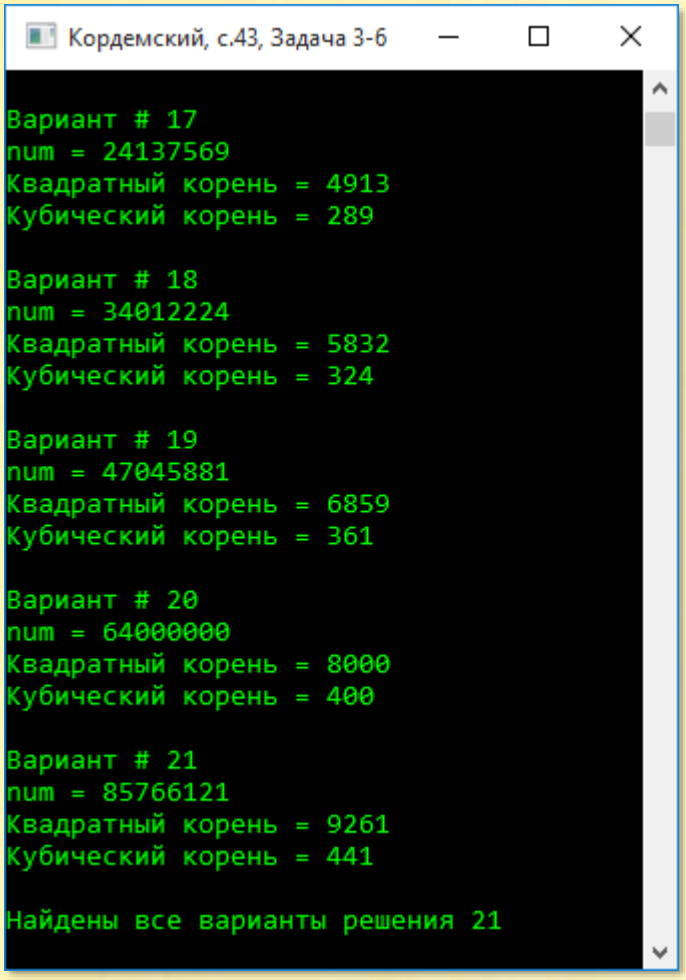
```
// ОПРЕДЕЛЯЕМ ЯВЛЯЕТСЯ ЛИ ЗАДАННОЕ ЧИСЛО
// ПОЛНЫМ КУБОМ
Function IsQube(num: int64): boolean;
begin
   var r := Trunc(Math.Round(Math.Pow(num, 1.0 / 3.0)));
   Result := r * r * r = num;
end;
```

Рисунок ниже показывает, что первое число с указанными в задаче свойствами, - 117 649, а второе – 7 529 536 – значительно больше 1 миллиона. А всего среди первого миллиарда чисел только 4 кратны 7 и являются полными квадратами и кубами.



Если слегка подправить функцию Solve, то можно найти ещё несколько любопытных чисел:

```
function Solve2(): integer;
begin
   Result := 0;
   var min := 1;
   var max := 100000000;

   for var num: int64 := min to max do
      . . .
end;
```



Или в более наглядной форме:

- $3375^2 = 225^3 = 11390625$
- $4913^2 = 289^3 = 24137560$
- И так далее.

Красивые цепочки равенств

Процедура без параметров
Бесконечный цикл ***for***
Тип данных ***int64***
Условный оператор ***if***
Оператор ***continue***
Оператор ***break***
Оператор ***exit***
Функция *с параметром*
Оператор приведения ***integer()***
Цикл ***while***

Задача 9 (14) из книги *Удивительный мир чисел* [КА86], страница 45:

Пусть символ Сц(N) выражает сумму цифр числа N в десятичной записи. Например, $Сц(5^3) = Сц(125) = 1 + 2 + 5 = 8$.

Мы утверждаем, что при одном и том же целом значении ***x*** имеют место две красивые цепочки тождеств:

1) $Сц(x^6) = Сц(x^7) = Сц(x^8) = Сц(x^{12}) = 6x$
2) $Сц(x^9) = Сц(x^{10}) = Сц(x^{11}) = Сц(x^{13}) = Сц(x^{17}) = Сц(x^{21}) = x^x$

Найдите подходящее значение *x*.

```
uses
   System;

// Кордемский, с.45, Задача 9

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.45, Задача 9';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
```

```
   Console.WriteLine('Красивые цепочки равенств');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Так как в условии задачи участвуют очень большие степени, то можно предположить, что число ***x*** должно быть, наоборот, очень маленьким.

В бесконечном цикле ***for*** мы перебираем значения ***x***, начиная с единицы, и при этом проверяем выполнение условий задачи:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var min := 1;
   var x := min - 1;

   while (true) do
   begin
      x += 1;

      var x6: int64 := x * x * x * x * x * x;
      if (Summa(x6) <> 6 * x) then continue;
      var x7: int64 := x6 * x;
      if (Summa(x7) <> 6 * x) then continue;
      var x8: int64 := x7 * x;
      if (Summa(x8) <> 6 * x) then continue;
      var x9: int64 := x8 * x;
      var x12: int64 := x6 * x6;
      if (Summa(x12) <> 6 * x) then continue;

      var xx: int64 := x;
      for var i := 2 to x do
         xx *= x;

      if (Summa(x9) <> xx) then continue;
```

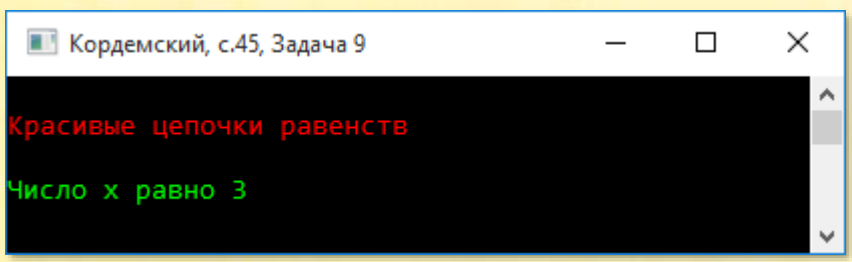
```
      var x10: int64 := x9 * x;
      if (Summa(x10) <> xx) then continue;
      var x11: int64 := x10 * x;
      if (Summa(x11) <> xx) then continue;
      var x13: int64 := x12 * x;
      if (Summa(x13) <> xx) then continue;
      var x17: int64 := x7 * x10;
      if (Summa(x17) <> xx) then continue;
      var x21: int64 := x11 * x10;
      if (Summa(x21) <> xx) then continue;
```

Как только все равенства станут верными, мы прерываем цикл оператором *break* и печатаем найденное число:

```
      break;
    end;

    Console.WriteLine('Число x равно ' + x);

    Console.WriteLine();
end;
```

Для подсчёта суммы цифр чисел мы используем функцию Summa:

```
// НАХОДИМ СУММУ ЦИФР ЗАДАННОГО ЧИСЛА
function Summa(num: int64): integer;
begin
    var sum: int64 := 0;
    while (num > 0) do
    begin
        sum += num mod 10;
        num := num div 10;
    end;
    Result := integer(sum);
end;
```



Как мы и предполагали, значение ***x*** совсем небольшое.

«Избранные» числа

Функция без параметров
Цикл ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор целочисленного деления ***mod***
Функция с параметром
Цикл ***while***

Задача 12 из книги *Удивительный мир чисел* [КА86], страница 65:

> Есть 80 четырёхзначных и 800 пятизначных чисел, не оканчивающихся нулем, и таких, что если от любого из них вычтем 999 в случае четырёхзначного числа и 9999 в случае пятизначного числа, то всякий раз получим обращённое число, т. е. записанное теми же цифрами, но в обратном порядке. Какие это числа?
>
> Найдите способ быстрого вычисления суммы этих чисел (без привлечения компьютера).

Мы проигнорируем замечание в скобках и решим сначала задачу для 4-значных чисел:

```
uses
   System;

// Кордемский, с.65, Задача 12

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.65, Задача 12';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('«Избранные» числа');
   Console.ForegroundColor := ConsoleColor.Green;
```

```
   Console.WriteLine();

   var nVar := Solve();
   Console.WriteLine('Найдены все варианты решения ' +
                     nVar.ToString());

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Так как условие задачи запрещает числам оканчиваться нулём, то минимальное 4-значное число равно 1001, а максимальное – 9999:

```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   Result := 0;
   var min := 1001;
   var max := 9999;
   var minus := 999;
```

Перебираем все 4-значные числа в цикле *for*:

```
   for var num := min to max do
   begin
        //число не заканчивается нулём:
      if (num mod 10 = 0) then
         continue;
```

Вычитаем из каждого числа 999:

```
var num2 := num - minus;
```

И сравниваем его с обращённым исходным числом:

```
if (num2 <> ReverseNum(num)) then
    continue;
```

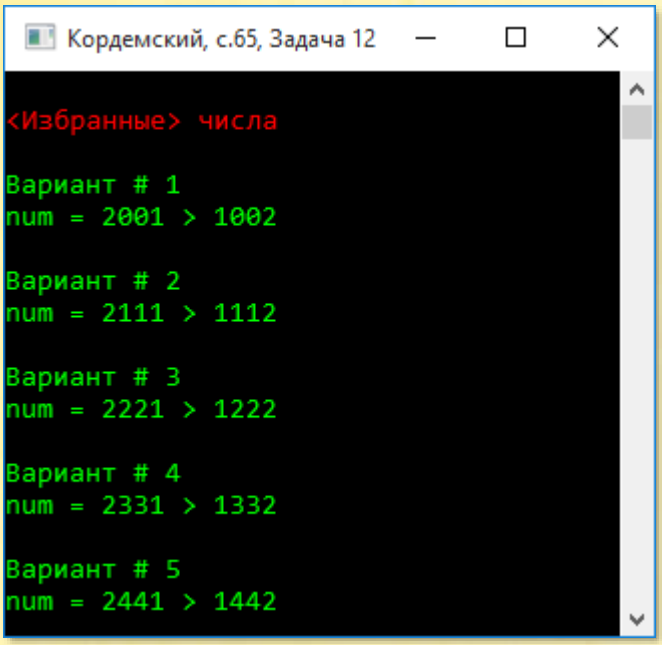
Если они совпадают, то найдено ещё одно решение задачи:

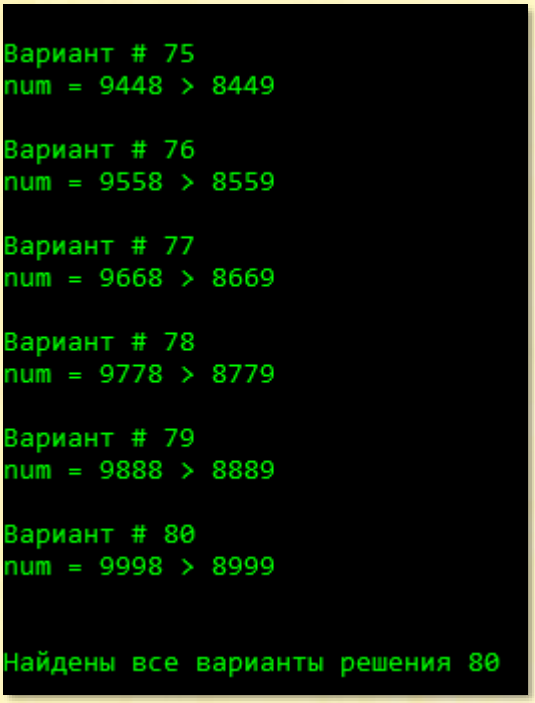
```
      Result += 1;
      Console.WriteLine('Вариант # ' +
                        result.ToString());
      var s := 'num = ' + num.ToString();
      s += ' > ' + num2.ToString();
      Console.WriteLine(s);
      Console.WriteLine();
   end
end;
```

Для обращения чисел мы используем функцию ReverseNum:

```
function ReverseNum(num: integer): integer;
begin
   var rev := 0;
   while (num > 0) do
   begin
      rev := rev * 10 + num mod 10;
      num := num div 10;
   end;
   Result := rev;
end;
```

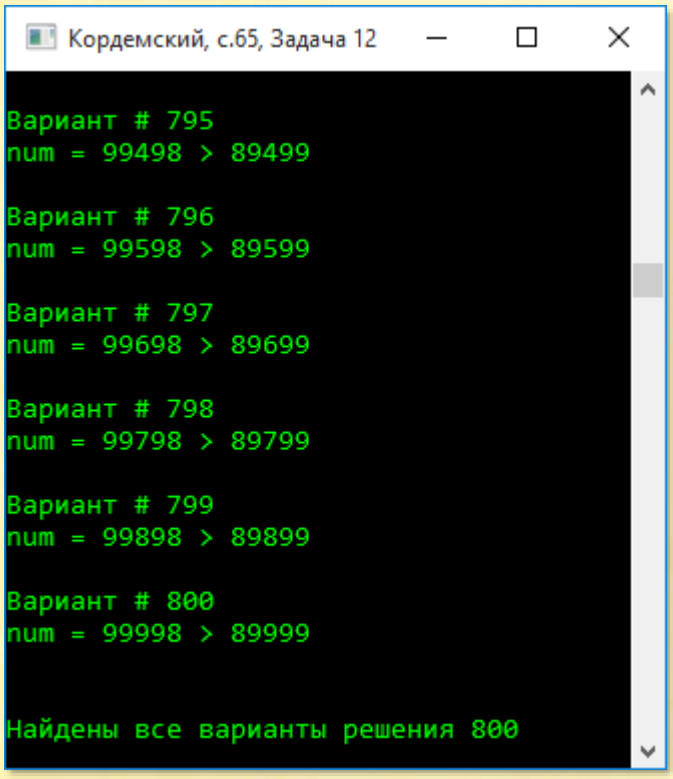
Как и указано в книге *Удивительный мир чисел*, всего существует 80 таких чисел:





Для поиска 5-значных чисел необходимо изменить значения локальных переменных в функции *Solve*:

```
//var min := 1001;
//var max := 9999;
//var minus := 999;

var min := 10001;
var max := 99999;
var minus := 9999;
```



5-значных чисел ровно в 10 раз больше, чем 4-значных:

Поскольку мы решаем задачу на компьютере, то было бы странно, если бы мы принялись сумму найденных чисел вычислять вручную! Поэтому объявляем переменную для подсчёта суммы:

```
// сумма чисел:
var sum := 0;
```

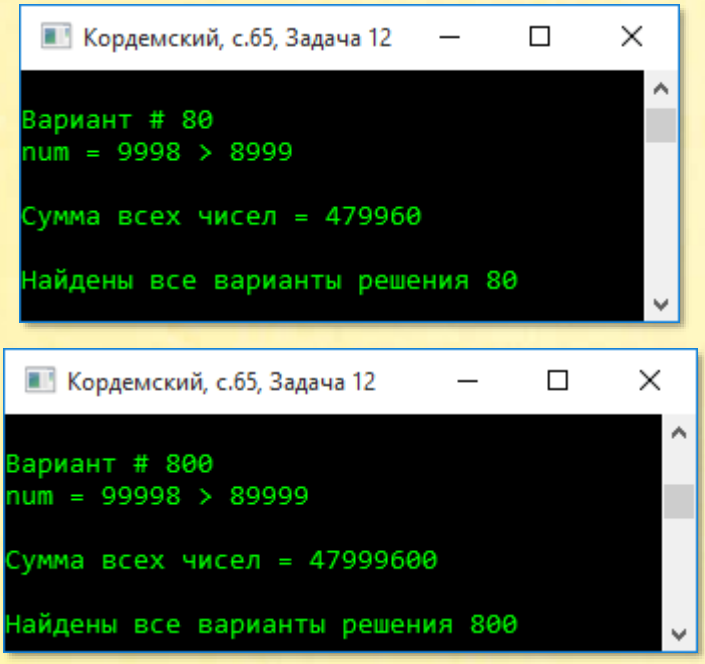
И каждое найденное число добавляем к общей сумме:

```
Result += 1;
sum += num;
```

В конце функции *Solve* печатаем эту сумму:

```
    end;
    Console.WriteLine('Сумма всех чисел = ' + sum);
    Console.WriteLine();
end;
```

Вот решение задачи для 4- и 5-значных чисел:



Безошибочный прогноз

Метод ***ArrRandom***
Процедура *с параметром*
Условный оператор ***if***
Оператор ***exit***
Операторы ***or*** *и* ***and***
Массив ***array of integer***
Цикл ***for***
Цикл ***do-while***
Метод ***Math.Abs***
Оператор цикла ***foreach***

Задача 16 (17) из книги *Удивительный мир чисел* [КА86], страница 67:

Расположите по кругу 4 произвольных натуральных числа a_1, a_2, a_3, a_4. Замените их абсолютными значениями разностей (Рис. 6.11):

$$|a_1 - a_2|, |a_2 - a_3|, |a_3 - a_4|, |a_4 - a_1|.$$

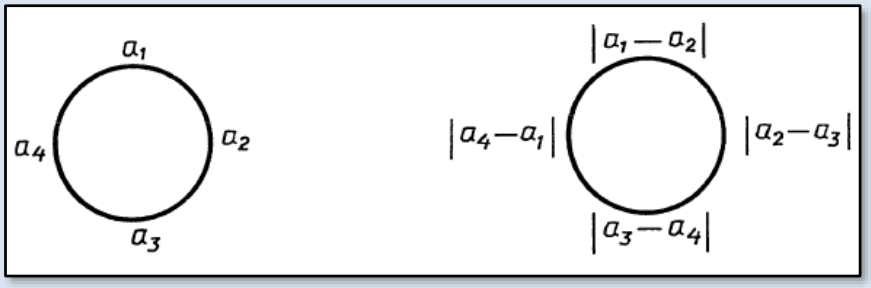


Рис. 6.11. Иллюстрация к задаче

С получившимися разностями поступите так же, как с исходными числами. Повторите процедуру вычисления разностей несколько раз, и на некотором шаге все разности одновременно станут нулями.

Можете начать вычисления не с 4, а с 8 или с 16, вообще с 2^k (k = 2, 3, ...) чисел - финал будет таким же.

Докажите хотя бы для 4 исходных чисел, что прогноз безошибочен.

Дополнительную информацию об этой задаче смотрите на странице 6.

Задача очень интересная, и мы за неё возьмёмся!

В главном блоке программы мы передаём процедуре *Solve* количество чисел, которые мы желаем испытать:

```
uses
   System;

// Кордемский, с.67, Задача 16

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.67, Задача 16';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Безошибочный прогноз');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve(4);

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

По условию задачи, это число не меньше 4 и должно быть степенью двойки, поэтому, прежде всего, следует проверить значение параметра n:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve(n: integer);
begin
   // число n должно быть степенью двойки:
   if ((n < 4) or (n and (n - 1) <> 0)) then
      exit;
```

Если условие задачи нарушено, то следующую часть кода выполнять нельзя, и мы возвращаемся в главный блок программы.

Если все числа изначально равны нулю, то условие задачи будет выполнено, но, скорее всего, нам потребуется сделать для этого несколько шагов, которые мы, ради интереса, подсчитываем в переменной **step**:

```
// номер шага:
var step := 0;
```

В задаче могут участвовать не только 4, но и 8, и 16 чисел, поэтому для их хранения потребуется **массив**, а не *n* отдельных переменных:

Число элементов в массиве нужно задать на 1 больше, чем число *n*. Как вы помните, нумерация элементов в массиве начинается с нуля, а в задаче индексы чисел начинаются с 1. Таким образом, в массиве **a** окажутся элементы с индексами *0..n*, причём нулевой элемент в задаче не используется, но мы знаем, что его значение равно 0.

Мы создаём массив *случайных* чисел, что соответствует присвоению значений числам $a_1..a_4$ в задаче:

```
// массив чисел:
var a := ArrRandom(n + 1, 0, 20);
a[0] := 0;
```

Тут же можно полюбопытствовать, что у нас получилось:

```
Console.WriteLine('Шаг {0}:', step);
// печатаем массив:
a.Println();
```

В задаче утверждается, что через некоторое число шагов все элементы массива **одновременно** станут нулями. На самом деле это не так: некоторые элементы

массива могут обратиться в нули раньше других, поэтому мы будем контролировать процесс по сумме всех элементов в массиве. Как только сумма обратится в нуль, цикл *repeat-until* будет закончен:

```
   // сумма элементов массива:
   var sum := 0;
   // заменяем элементы массива
   // абсолютными значениями разностей:
   repeat
      var a1 := a[1];
      for var i := 1 to n - 1 do
         a[i] := Math.Abs(a[i] - a[i + 1]);

      a[n] := Math.Abs(a[n] - a1);
      step += 1;
      Console.WriteLine('Шаг {0}:', step);
      a.Println();
      // находим сумму элементов массива:
      sum := 0;
      foreach var i in a do
         sum += i;
   until not (sum <> 0);
   Console.WriteLine();
end;
```

Важно учесть, что последнюю разность

$$a_4 - a_1$$

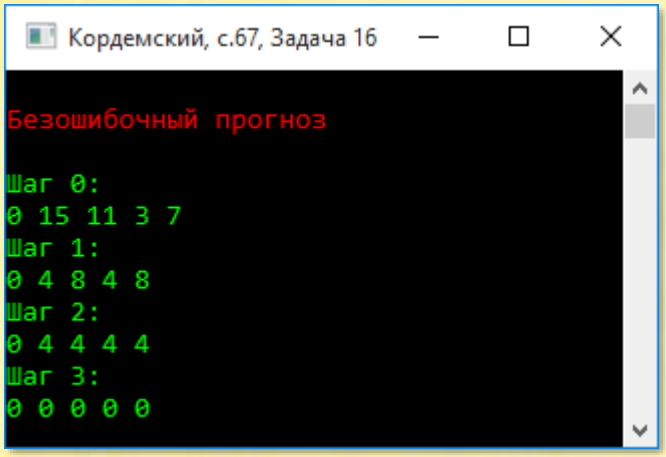
нельзя вычислить в цикле *for*. Действительно, значение первого элемента массива будет заменено абсолютным значением разности

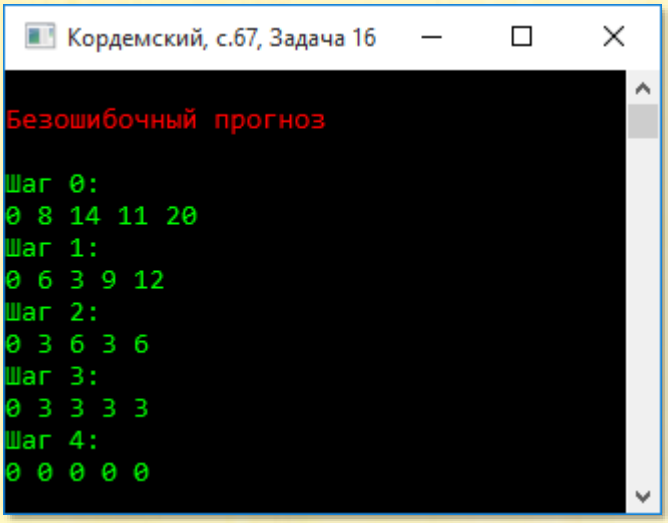
$$a_1 = a_1 - a_2$$

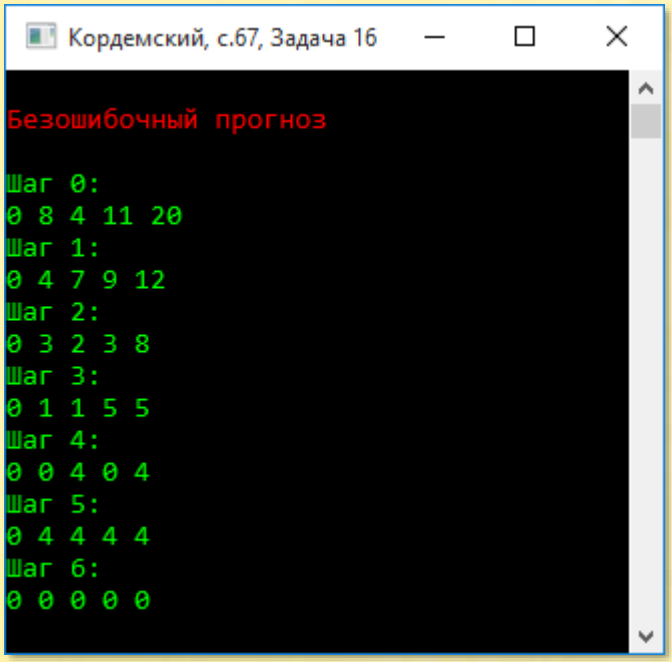
и последняя разность будет вычислена неверно!

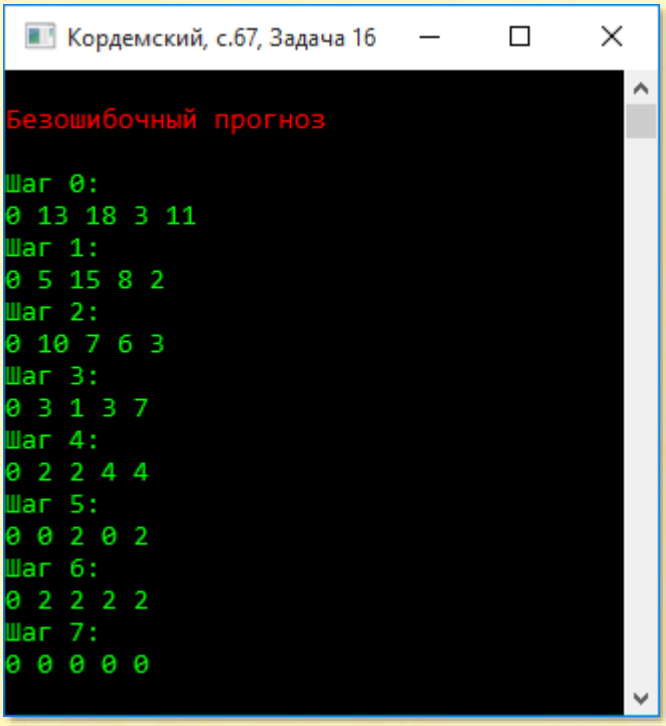
Поэтому мы перед началом цикла изменения значений элементов массива запоминаем во вспомогательной переменной `a1` текущее значение первого элемента. Его мы и используем для замены значения последнего элемента массива.

Результаты проверки «гипотезы» для четырёх чисел показаны на рисунках:









Как вы видите, для обнуления элементов массива может потребоваться разное число шагов – в зависимости от начальных значений элементов массива.

С числами 8, 16 и далее поэкспериментируйте самостоятельно!

Ошибочный прогноз

Процедура без параметров
Массив ***array of integer***
Цикл ***for***
Цикл ***repeat-until***
Метод ***Math.Abs***
Операторы ***and*** *и* ***or***

Пример из книги *Удивительный мир чисел* [КА86], страница 6:

Иной результат наблюдается для серии разностей в случае комплекта из трёх произвольных натуральных чисел: в финале всегда получаются две единицы и нуль в том или ином чередовании.

Пример. Пусть исходная тройка чисел $R_0 = (7, 12, 1)$. Тогда последовательность разностей будет:

$R_1 = (5, 11, 6)$
$R_2 = (6, 5, 1)$
$R_3 = (1, 4, 5)$
$R_4 = (3, 1, 4)$
$R_5 = (2, 3, 1)$
$R_6 = (1, 2, 1)$
$R_7 = (1, 1, 0)$

Поскольку один пример ничего не доказывает, то мы напишем программу и подвергнем серьёзному сомнению и проверке это утверждение.

```
uses
   System;

// Кордемский, с.6, Пример

begin
```

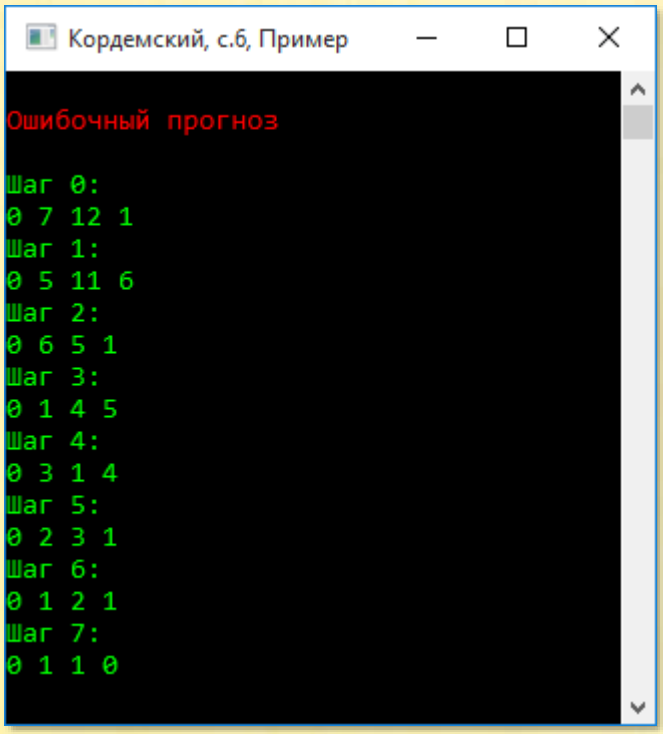
```
    // заголовок окна:
    Console.Title := 'Кордемский, с.6, Пример';
    Console.WriteLine('');
    Console.ForegroundColor := ConsoleColor.Red;
    Console.WriteLine('Ошибочный прогноз');
    Console.ForegroundColor := ConsoleColor.Green;
    Console.WriteLine();

    Solve();

    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```

Небольшие изменения в процедуре Solve – и можно приниматься за научно-исследовательские работы!

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
    // тройка чисел:
    var n := 3;

    // номер шага:
    var step := 0;
    // массив чисел:
    var a := ArrRandom(n + 1, 0, 20);
    a[0] := 0;

    //a[1] := 7;
    //a[2] := 12;
    //a[3] := 1;

    Console.WriteLine('Шаг {0}:', step);
    // печатаем массив:
    a.Println();

    // сумма элементов массива:
    var sum := 0;
    var flg := false;
    // заменяем элементы массива
    // абсолютными значениями разностей:
```

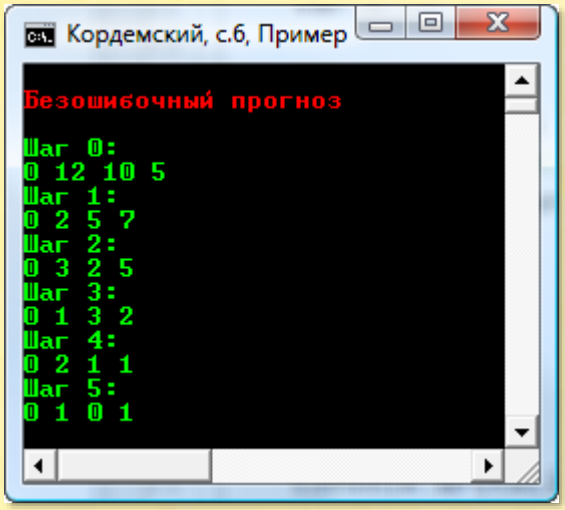
```
    repeat
      var a1 := a[1];
      for var i := 1 to n - 1 do
        a[i] := Math.Abs(a[i] - a[i + 1]);

      a[n] := Math.Abs(a[n] - a1);
      step += 1;
      Console.WriteLine('Шаг {0}:', step);
      a.Println();

        if ((a[1] * a[2] * a[3] = 0) and
          ((a[1] - a[2] - a[3] = 0) or (a[2] = a[3]))) then
          flg := true;

    until not (not flg);
    Console.WriteLine();
end;
```

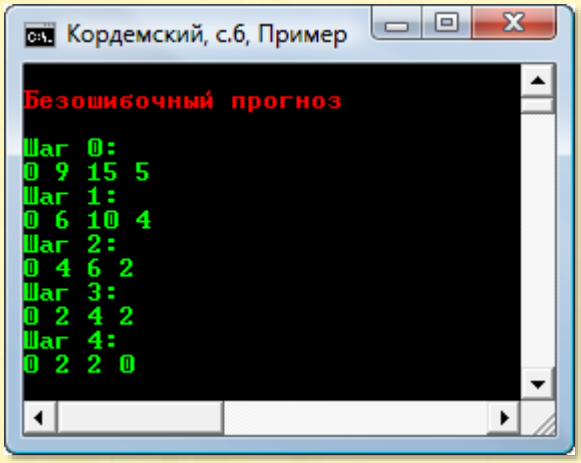
Раскомментируйте строчки с условиями примера и запустите программу. Рисунок подтверждает, что на 7 шаге одно число превратится в нуль, а остальные два – в единицу:

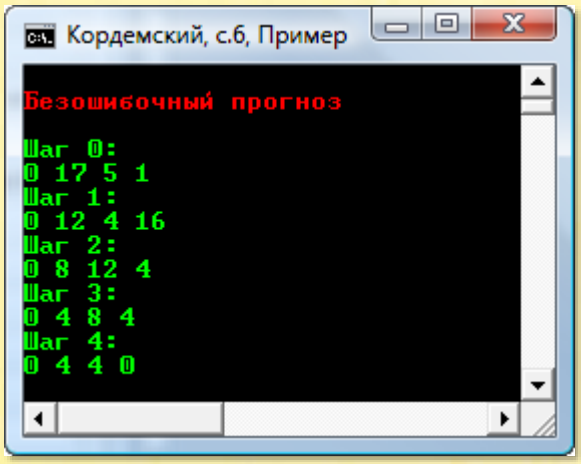


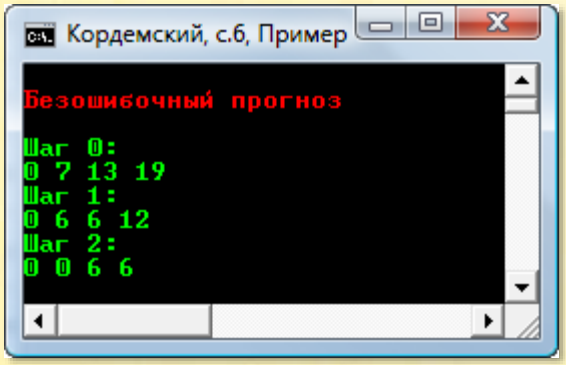
Теперь закомментируйте раскомментированное и пуститесь в свободное плавание по числовому океану!

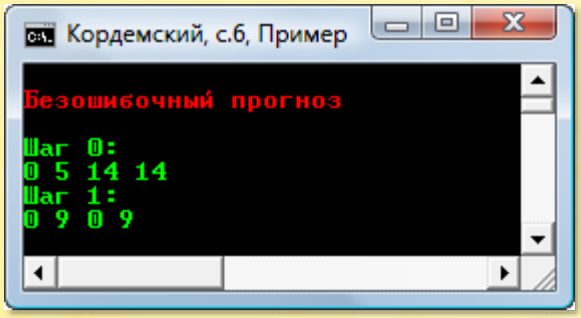
Сначала было тихо и штильно (рисунок слева), но вскоре заштормило (рисунок правее). И не на шутку (рисунок ниже и ещё ниже)!

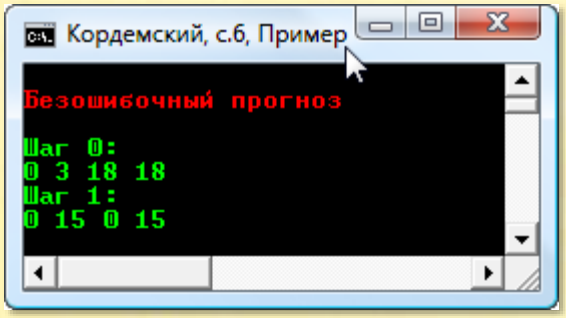








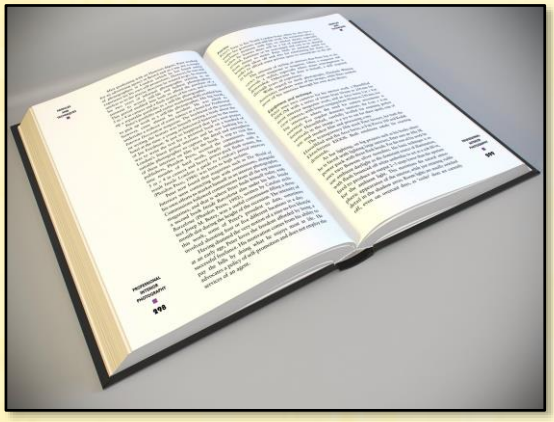




Как вы видите, элементы массива **a** всегда превращаются в элегантные числа: 1 нуль и «ненули». Причём ненули это не всегда пара единиц!

Вывод: *Не говори гоп, пока не проверишь решение на компьютере!*

Нумерация страниц



***Процедура** без параметров*
*Цикл **for***
*Вложенные условные операторы **if-else***

Задача 6 из книги *Удивительный мир чисел* [КА86], страница 64:

Я спросил наборщика типографии:

- Сколько отдельных литер с цифрами потребуется для нумерации 1000 страниц книги?

- Легко вычислить,- ответил наборщик. - Вот моё решение:

3000 - 18 – 90 +1= 2893.

Я решил иначе, но получил такой же результат.

Придумайте свой план решения и найдите объяснение способу, предложенному наборщиком.

```
uses
   System;

// Кордемский, с.64, Задача 6

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.64, Задача 6';
```

```
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Нумерация страниц');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

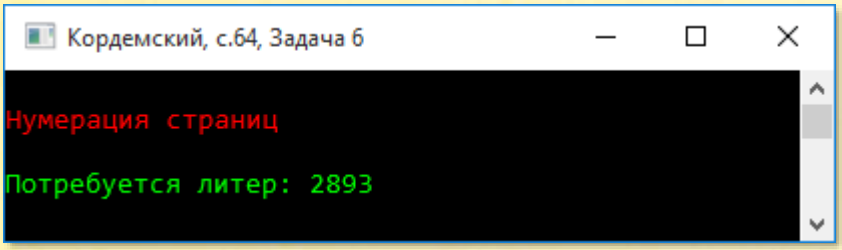
Число литер легко найти так:

- для записи чисел 1..9 нужно по 1 литере
- для записи чисел 10..99 нужно по 2 литеры
- для записи чисел 100..999 нужно по 3 литеры
- для записи чисел 1000..9999 нужно по 4 литеры

В процедуре Solve мы «пролистываем» книгу от 1 до 1000 страницы и считаем литеры, как описано выше:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   // число литер:
   var sum := 0;
   for var i := 1 to 1000 do
   begin
      if (i < 10)  then sum += 1
      else if (i < 100) then sum += 2
      else if (i < 1000) then sum += 3
      else sum += 4;
   end;
   var s := 'Потребуется литер: ' + sum;
   Console.WriteLine(s);
   Console.WriteLine();
end;
```

Все правы: потребуется 2893 литеры:



Сколько страниц в книге?

Процедура без параметров
Константы
Бесконечный цикл **while**
Вложенные условные операторы **if-else**
Оператор **break**

Задача 5 из книги *Удивительный мир чисел* [КА86], страница 64:

> Для нумерации страниц книги наборщик типографии использовал 2529 литер с цифрами. На каждой литере одна цифра.
>
> Сколько страниц в этой книге?

Задача очень похожа на предыдущую, поэтому нам нужно только подправить её код:

```
uses
   System;

// Кордемский, с.64, Задача 5

begin
```

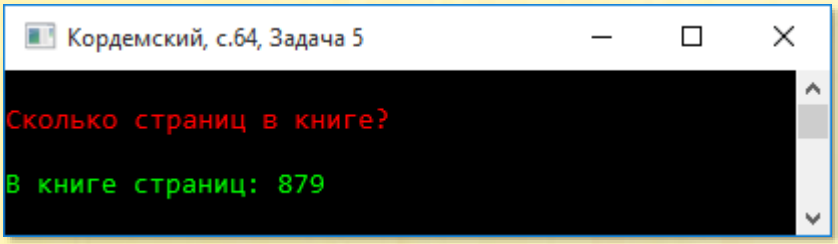
```
   // заголовок окна:
   Console.Title := 'Кордемский, с.64, Задача 5';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Сколько страниц в книге?');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
// число использованных литер:
const
   SUMMA = 2529;
begin
   // число литер:
   var sum := 0;
   var i := 1;
```

Мы знаем, что первая страница обозначена числом 1, вторая – числом 2, и так далее. Если мы будем последовательно находить сумму литер, затраченных на нумерацию страниц, то рано или поздно она сравняется с заданной – SUMMA (а если задача составлена неверно, то это событие может и не произойти!). На этом бесконечный цикл *for* заканчивается, и мы печатаем ответ на задачу:

```
   // перелистываем страницы и считаем литеры:
   while (true) do
   begin
      if (i < 10)  then sum += 1
      else if (i < 100)  then sum += 2
      else if (i < 1000) then sum += 3
      else sum += 4;
      if (sum = SUMMA) then
      begin
         var s := 'В книге страниц: ' + i;
```

```
            Console.WriteLine(s);
            break;
        end
        else if (sum > SUMMA) then
        begin
            var s := 'Задача решения не имеет!';
            Console.WriteLine(s);
            break;
        end;
        i += 1;
    end;
    Console.WriteLine();
end;
```

В книге оказалось 879 страниц, а задачу автор составил верно:



Трёхзначное число

Процедура без параметров
Цикл ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор целочисленного деления ***mod***
Оператор деления ***div***

Задача 3 из книги *Удивительный мир чисел* [КА86], страница 63:

Найдите трёхзначное число, обладающее следующими свойствами:

- число десятков на 4 меньше числа единиц, но на 4 больше числа сотен;
- если цифры этого числа разместить в обратном порядке, то новое полученное число будет на 792 больше искомого.

```
uses
   System;

// Кордемский, с.63, Задача 3

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.63, Задача 3';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Трёхзначное число');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Чтобы найти трёхзначное число с требуемыми свойствами, необходимо перебрать все трёхзначные числа и выбрать те из них, которые удовлетворяют условиям задачи.

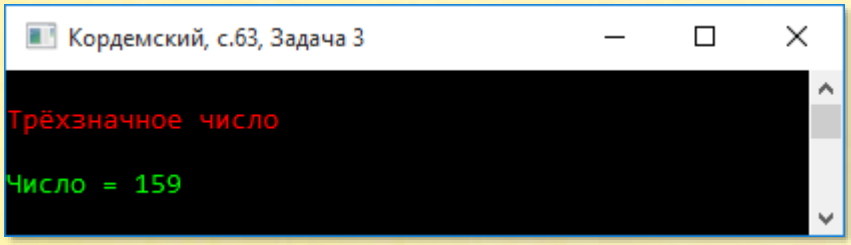
```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var min3 := 100;
   var max3 := 999;

   for var n := min3 to max3 do
   begin
```

Прежде всего, нужно найти, сколько единиц, десятков и сотен содержит текущее число. Это нетрудно сделать с помощью двух операций – целочисленного деления и деления с остатком:

```
// число единиц:
var e := n mod 10;
// число десятков:
var d := n div 10 mod 10;
if (d <> e - 4) then
   continue;
// число сотен:
var s := n div 100 mod 10;
if (d <> s + 4) then
   continue;
```

Выполнить проверки и того проще. И вот мы уже получаем ответ на эту задачу:

```
      var str := 'Число = ' + n;
      Console.WriteLine(str);
      Console.WriteLine();
   end
end;
```



Поскольку мы перебрали все трёхзначные числа и нашли только одно, которое удовлетворяет условиям задачи, то последняя проверка на разность с перевёрнутым числом оказалась бы лишней!

Таких чисел только два

Процедура без параметров
Цикл ***for***
Оператор целочисленного деления ***mod***
Оператор деления ***div***
Условный оператор ***if***
Оператор ***continue***

Задача 1 из книги *Удивительный мир чисел* [КА86], страница 63:

> Есть только два двузначных числа, каждое из которых равно неполному квадрату разности своих цифр.
>
> Найдите эти числа.
>
> Чтобы облегчить решение, подскажем, что одно число на 11 больше другого.

Так как двузначных чисел очень мало, то мы и без подсказки найдём оба искомых числа!

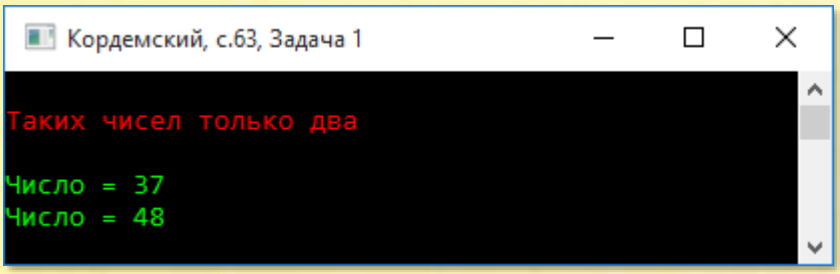
```
uses
   System;

// Кордемский, с.63, Задача 1

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.63, Задача 1';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Таких чисел только два');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();
```

```
   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Выделяем из каждого числа его единицы и десятки и сравниваем само число с неполным квадратом разности его цифр:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var min2 := 10;
   var max2 := 99;
   for var n := min2 to max2 do
   begin
      // число единиц:
      var e := n mod 10;
      // число десятков:
      var d := n div 10 mod 10;
      if (e * e + d * d - e * d <> n) then
         continue;

      var str := 'Число = ' + n;
      Console.WriteLine(str);
   end;
   Console.WriteLine();
end;
```

Задача решена:



Ещё два числа

Процедура без параметров
Условный оператор ***if***
Цикл ***for***
Оператор ***continue***
Оператор деления ***div***
Оператор целочисленного деления ***mod***

Задача 2 из книги *Удивительный мир чисел* [КА86], страница 63:

> Сходным свойством обладают ещё два двузначных числа: каждое равно неполному квадрату суммы своих цифр.
>
> Найдите эти числа, зная, что одно число на 50 больше другого.

Эта задача решается точно так же, как предыдущая:

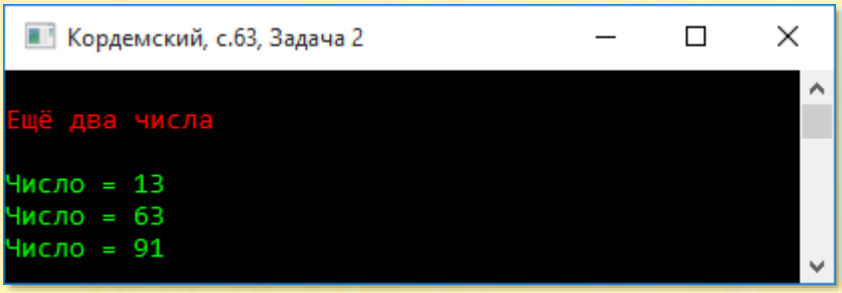
```
uses
   System;

// Кордемский, с.63, Задача 2

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.63, Задача 2';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Ещё два числа');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
```

```
end.
```

Только в процедуре Solve нужно исправить одну строчку:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   var min2 := 10;
   var max2 := 99;
   for var n := min2 to max2 do
   begin
      // число единиц:
      var e := n mod 10;
      // число десятков:
      var d := n div 10 mod 10;
      if (e * e + d * d + e * d <> n)    then
         continue;

      var str := 'Число = ' + n;
      Console.WriteLine(str);
   end;
   Console.WriteLine();
end;
```

Правда, к нашему удивлению чисел оказалось не два, а три:



Да, с полным перебором состязаться невозможно!

Отгадать число, ничего не спрашивая

Процедура без параметров
Вложенные циклы ***for***
Условный оператор ***if***
Оператор ***continue***
Оператор деления ***div***
Оператор целочисленного деления ***mod***

Ради счастья, ради нашего,
Если хочешь ты его,
Ни о чем меня не спрашивай,
Не расспрашивай, не выспрашивай,
Не выведывай ничего.

Песня из комедии *Свадьба в Малиновке*

Задача 4 из книги *Удивительный мир чисел* [КА86], страница 36:

Пусть кто-нибудь задумает двузначное число в виде 10x + y, где **x** — цифра десятков, **y** - цифра единиц, x – y ≥ 2, y ≠ 0. Потребуйте теперь, чтобы он переставил цифры в обратном порядке и вычел меньшее число из большего. Полученную разность пусть сложит с нею же, но написанной в обратном порядке следования цифр. Ничего не спрашивая у загадавшего, вы сообщаете, что у него получилось 99.

Пример. Пусть задумано 75. Загадавший должен выполнить следующие действия: 75 - 57=18, 18 + 81=99.

В общем виде: пусть задумано 10x + y, где x – y ≥ 2. При обратном расположении цифр число имеет вид 10y + x.

Абсолютная величина разности:

$R = |10x + y - (10y + x)| = 9 \cdot |x - y|$.

Так как x – y ≥ 2, то $R = 9 \cdot (x - y) = 10 \cdot (x - y) - (x - y) + 10 - 10$.

То есть разность можно представить в виде:

$R = 10(x - y - 1) + (10 - (x - y))$. (1)

Запишем число с обратным расположением цифр:

$10(y - x + 10) + (x - y - 1)$ (2)

Сложив (1) и (2), получим:

$10(x - y - 1) + (10 - (x - y)) + 10(y - x + 10) + (x - y - 1) = 99$.

Итак, независимо от выбора цифр x и y двузначного числа при $x - y \geq 2$ и $y \neq 0$ всегда получается число 99.

Если вышеуказанные действия будут применены к любому **трёхзначному** числу $100x + 10y + z$ при соблюдении условий $x - y > 2$, $x - y = y - z$, то в результате всегда получится 1089.

Для **четырёхзначного** числа $1000x + 100y + 10z + t$ при условии $x > y > z > t > 0$, $x - y = y - z = z - t$ в результате всегда получится число 10 890 и т. д.

В книге это не задача, а **фокус**, но мы обойдёмся без фокусов! Наша задача: с помощью компьютера и полного перебора подтвердить (или опровергнуть) доказательство, приведённое в книге.

Из условия задачи следует, что цифра ***x*** изменяется в диапазоне 3..9, а цифра ***y*** – в диапазоне 3..7, при этом цифра ***x*** как минимум на двойку больше цифры ***y***.

```
uses
   System;

// Кордемский, с.36, Задача 4

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.36, Задача 4';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Отгадать число, ничего не спрашивая');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
```

```
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   for var x := 3 to 9 do
      for var y := 1 to 7 do
      begin
         if (x - y < 2) then
            continue;
```

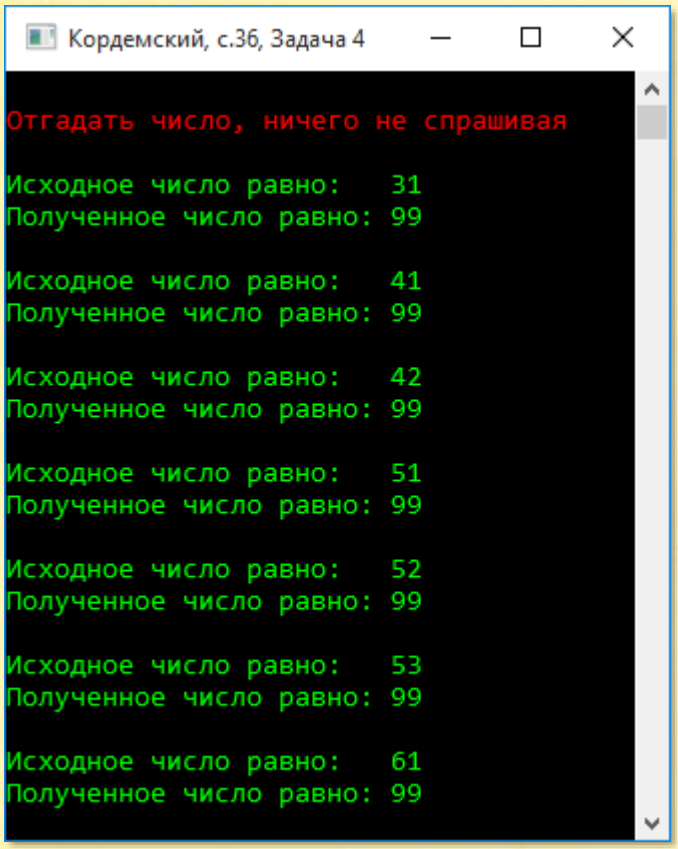
Получив пару верных цифр, мы составляем из них число, переворачиваем его, находим абсолютную разность прямого и обратного чисел, а затем - сумму полученной разности и её «зеркального отражения»:

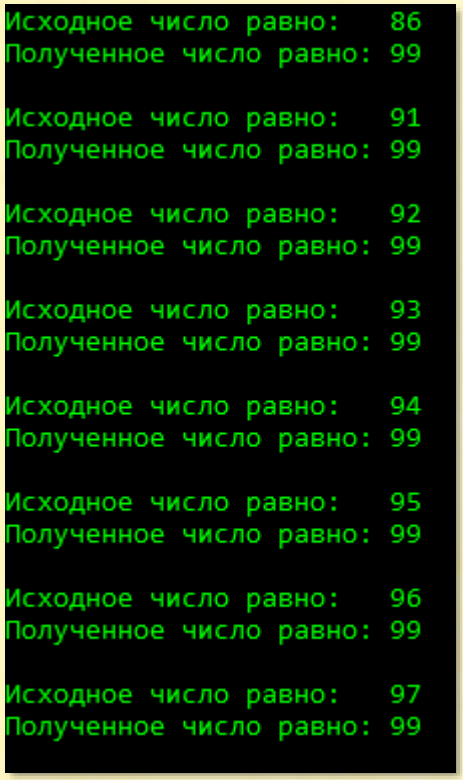
```
         // прямое число:
         var num := 10 * x + y;
         // обратное число:
         var numr := 10 * y + x;
         // разность чисел:
         var r := Abs(num - numr);
         // число единиц в разности:
         var e := r mod 10;
         // число десятков в разности:
         var d := r div 10;
         // сумма:
         var sum := r + e * 10 + d;
```

Для контроля печатаем каждое двузначное число, удовлетворяющее условиям задачи, а также результат наших действий над ним:

```
         Console.WriteLine('Исходное число равно:   ' + num);
         Console.WriteLine('Полученное число равно: ' + sum);
         Console.WriteLine();
      end
end;
```

Часть списка представлена на рисунках, на которых видно, что **все двузначные числа обращаются в 99**:





Трёх- и четырёхзначные числа проверьте самостоятельно!

Три лягушки

Процедура без параметров
Константы
Массив integer[]
Массив boolean[]
Бесконечный цикл for
Вложенные циклы for
Условный оператор if
Оператор break
Оператор continue
Комбинированные операторы присваивания

Задача 6 из книги *Удивительный мир чисел* [КА86], страница 51:

Три лягушки находятся на дне колодца глубиной 60 м. За день они поднимаются на 18 м каждая, а потом спускаются первая на 12 м, вторая на 16 м, третья на 17 м и остаются на своих местах до следующего дня. На следующий день каждая лягушка проделывает снова такой же маршрут и т. д.

Через сколько дней лягушки выйдут из колодца?

```
uses
   System;

// Кордемский, с.51, Задача 6

begin
   // заголовок окна:
   Console.Title := 'Кордемский, с.51, Задача 6';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Три лягушки');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();
```

```
   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Для начала объявим необходимые константы и массивы:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
const
   // глубина колодца:
   GLUBINA = 60;
   // высота подъёма:
   UP = 18;
begin
   // глубина опускания:
   var DOWN := new integer[](12, 16, 17 );
   // текущее расстояние до поверхности:
   var frogs := new integer[](GLUBINA, GLUBINA, GLUBINA);
   // массив освободившихся лягушек:
   var free := new boolean[](false, false, false);
```

Приключения начинаются! По значению элементов массива free мы определяем, все ли лягушки-квакушки выбрались из колодца. Если это радостное событие состоялось, значит, все элементы в массиве обратились в *true*:

```
   var n := 1 - 1;
   while (true) do
   begin
      n += 1;
      // все лягушки свободны:
      var flg := true;
      for var f := 0 to frogs.Length - 1 do
      begin
         flg := flg and free[f];
      end;
```

```
      if (flg) then
      begin
         Console.WriteLine();
         Console.WriteLine('Все лягушки свободны!');
         Console.WriteLine();
         break;
      end;
```

На этом решение задачи заканчивается, и мы с чистой совестью отпускаем их на свободу, а бесконечный цикл *for* прерываем оператором *break*.

Каждый день лягушки поднимаются на UP метров:

```
// лягушки поднялись вверх:
for var f := 0 to frogs.Length - 1 do
begin
   // эта лягушка уже свободна:
   if (free[f]) then continue;
   frogs[f] -= UP;
```

Если текущее значение лягушки в массиве frogs не больше нуля, значит, лягушка выкарабкалась из колодца:

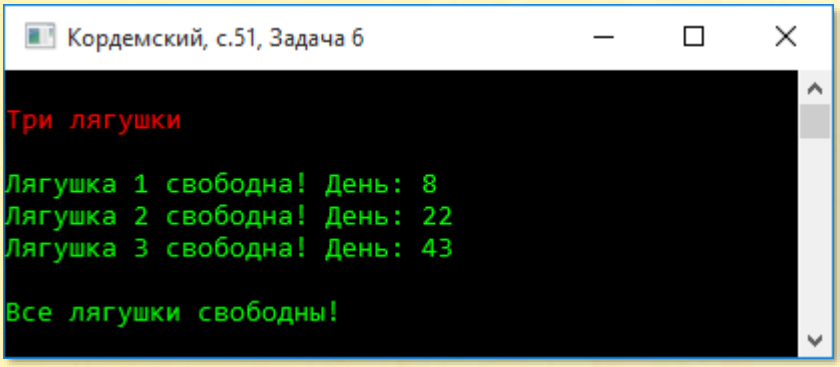
```
   // лягушка выбралась из колодца?
   free[f] := frogs[f] <= 0;
   if (free[f]) then
      Console.WriteLine('Лягушка {0} свободна! День: {1}', f + 1,
n);
end;
```

Если лягушкам выбраться не удалось, то они опускаются вниз – каждая на свой «размер»:

```
      // лягушки опускаются:
      for var f := 0 to frogs.Length - 1 do
```

```
        begin
            // эта лягушка уже свободна:
            if (free[f]) then continue;
            frogs[f] += DOWN[f];
        end
    end;
    Console.WriteLine();
end;
```

Запускаем программу и освобождаем лягушек:



Ура-ква-ква! Квак преквасен этот мир!

Сумма пятизначных чисел

Процедура без параметров
Массив ***array[,] of integer***
Вложенные циклы ***for***

Задача 11 из книги *Удивительный мир чисел* [КА86], страница 65:

> Предположим, что из цифр 1, 2, 3, 4, 5, 6 составлены всевозможные пятизначные числа, причём все цифры в записи каждого числа различны.
>
> **Чему равна сумма** всех таких пятизначных чисел?

Легко посчитать, что пятизначных чисел из цифр *1..6* можно составить столько же, сколько и перестановок из этих цифр.

Первое место в пятизначном числе может занимать одна из 6 цифр, второе – одна из 5 оставшихся, и так далее. Всего:

$$6 * 5 * 4 * 3 * 2 = 6!$$

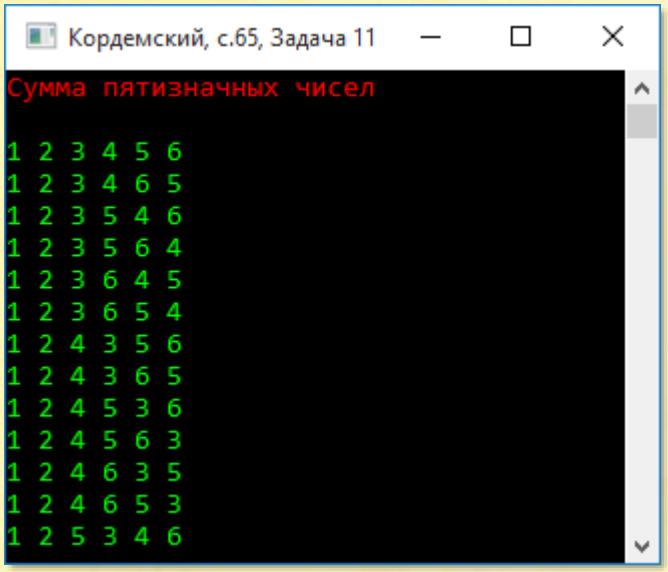
Следовательно, мы должны сначала получить все перестановки из цифр *1..6*, затем составить из них пятизначные числа, и наконец, найти их сумму.

Все перестановки генерирует функция *Permutation*, которая и возвращает их в массиве **perms**:

```
uses
   System;

// Кордемский, с.65, Задача 11

// ВЫЧИСЛЯЕМ ФАКТОРИАЛ
function Factorial(num: integer): integer;
begin
```

```
   var fact := 1;
   for var i := 1 to num do
      fact *= i;

   Result := fact;
end;

// ГЕНЕРИРУЕМ ПЕРЕСТАНОВКИ
function Permutation(n: integer): array[,] of integer;
begin
   // число перестановок:
   var AllPerm := Factorial(n);
   / /массив перестановок:
   var perms := new integer[AllPerm, n];

   // текущее число перестановок:
   var nPerm := 0;

   // массив чисел:
   var a := new integer[n + 2];
   // начальная перестановка:
   for var e := 1 to n do
      a[e] := e;

   var j: integer;
   repeat
      for var id := 1 to n do
         perms[nPerm, id - 1] := a[id];

      nPerm += 1;
      var i := n;
      while (a[i - 1] > a[i]) do
         i -= 1;
      j := i - 1;
      var h := a[j];

      while (a[i + 1] > h) do
         i += 1;
      a[j] := a[i];
      a[i] := h;
      i := j + 1;
      var k := n;
      while (i < k) do
      begin
         h := a[i];
```

```
            a[i] := a[k];
            a[k] := h;
            i += 1;
            k -= 1;
        end
    until not (j <> 0);

    Result := perms;
end;
```



Поскольку перестановки состоят из 6 цифр, а нам нужны только 5, то мы будем составлять 5-значные числа из первых пяти цифр (или из последних пяти цифр – кто как пожелает).

Зная цифры, само число легко найти, последовательно умножая число, составленное из *1..n-1* первых цифр на 10 и прибавляя к произведению следующую цифру:

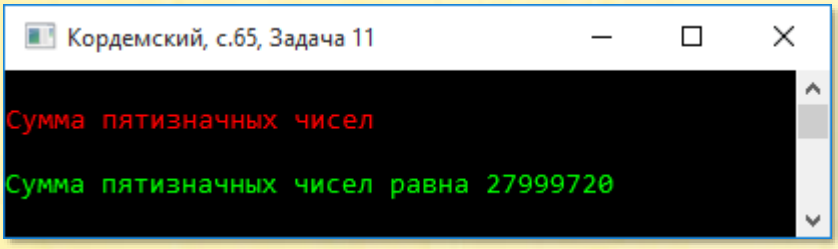
```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
    // генерируем перестановки:
    var perms := Permutation(6);

    // находим сумму всех чисел:
    var sum := 0;
```

```
    for var p := 0 to Factorial(6) - 1 do
    begin
      var num := 0;
      for var n := 0 to 4 do
      begin
        num := num * 10 + perms[p, n];
      end;
```

Полученное 5-значное число добавляем к промежуточной сумме:

```
      sum += num;
    end;
    Console.WriteLine('Сумма пятизначных чисел равна ' + sum);

    Console.WriteLine();
end;
```

Запускаем программу и получаем ответ:



Премия за изобретение

*Массив **integer[]***
Константы
***Функция** без параметров*
***Функция** с параметрами*
*Оператор **return***
***Функция** с параметрами-массивами*
*Цикл **for***

Задача 15 (16) из книги *Удивительный мир чисел* [КА86], страница 67:

> Четыре изобретателя независимо друг от друга придумали 10 различных приспособлений: Андрей - 1, Борис - 2, Виктор - 3 и Григорий - 4. Когда внедрили их в производство, то выяснилось, что каждое приспособление, разработанное одним и тем же изобретателем, даёт одну и ту же годовую экономию:
>
> - чьё-то по 25 тыс. р.,
> - чьё-то по 125 тыс. р.,
> - чьё-то по 625 тыс. р.,
> - а чьё-то даже по 3125 тыс. р.
>
> В результате годовая экономия составила 8350 тыс. р.
>
> 10% этой суммы выделили на премию, которую распределили между изобретателями пропорционально вкладу каждого в общую годовую экономию.
>
> **Сколько рублей получил каждый из них в качестве премии?**

Создадим **массив** для хранения годовой экономии по приспособлениям каждого изобретателя:

```
// массив годовой экономии приспособлений:
var eco := new integer[](0, 25, 125, 625, 3125);
```

Чтобы используемые значения имели индексы *1..4*, мы добавляем нулевой элемент с нулевым же значением. Тогда:

```
eco[1] =   25
eco[2] =  125
eco[3] =  625
eco[4] = 3125
```

Для нахождения годовой экономии мы должны вычислить сумму годовых экономии для каждого изобретателя, но мы не знаем авторов изобретений.

Если предположить, что *eco[1]* принадлежит Андрею, то его годовая экономия составит *eco[1] * 1 = 25*.

Аналогично вычисляем для других изобретателей:

```
eco[2] * 2 =   250
eco[3] * 3 =  1875
eco[4] * 4 = 12500
```

Итого: 14650. Не совпадает с условием задачи, в которой указано, что годовая экономия составила 8350. Это значит, что изобретения распределились иначе!

Для удобства сведём все данные в **таблицу**:

eco[1]	eco[2]	eco[3]	eco[4]
25	125	625	3125
1	2	3	4
25	250	1875	12500

Из неё видно, что годовая экономия каждого изобретателя **n** зависит от годовой экономии каждого изобретения **eco[n]** и номера самого изобретателя (точнее, от числа его изобретений). Мы предположили, что изобретатель №1 внедрил приспособление с годовой экономией *eco[1] = 25*, и так далее.

Оказалось, что это не так. Теперь мы можем переставить **красные** числа в таблице иначе и снова посчитать общую годовую экономию:

eco[1]	eco[2]	eco[3]	eco[4]
25	125	625	3125
1	2	4	3
25	250	2500	9375

```
25 + 250 + 2500 + 9375 = 12150
```

Опять не получилось! Мы должны составить новую перестановку изобретателей, найти для неё годовую экономию – и так продолжать до тех пор, пока сумма зелёных чисел в таблице не совпадёт с заданной.

Поскольку алгоритм мы придумали, и главную роль в нём играют перестановки чисел *1..4*, то вполне разумно обратиться к функции Permutation, которая умеет генерировать перестановки. При этом мы должны учесть, что нам нужны не сами перестановки, а общая годовая экономия, так что этот метод нужно доработать.

Начнём новый проект с объявления константы SUMMA, значение которой равно общей годовой экономии:

```
uses
   System;

// Кордемский, с.67, Задача 15

// годовая экономия:
const
   SUMMA = 8350;
```

Главный блок вызывает функцию *Solve* для решения поставленной задачи, получает от него число найденных решений и печатает его на экране:

```
begin
   // заголовок окна:
   Console.Title := 'Премия за изобретение';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Премия за изобретение');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   var nVar := Solve();

   Console.WriteLine();
   Console.WriteLine('Найдены все варианты решения ' +
                     nVar.ToString());

   Console.WriteLine();
```

```
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Функция Solve создаёт массив *eco*, о котором мы говорили выше, и передаёт его функции *Permutation*:

```
// РЕШАЕМ ЗАДАЧУ
function Solve(): integer;
begin
   // массив годовой экономии приспособлений:
   var eco := new integer[](0, 25, 125, 625, 3125);
   Result := Permutation(4, eco);
end;
```

На этот раз функция Permutation получает не только число элементов в массиве, но и массив годовой экономии для приспособлений каждого изобретателя. Так мы учитываем специфику конкретной задачи:

```
// ГЕНЕРИРУЕМ ПЕРЕСТАНОВКИ
function Permutation(n: integer; eco: array of integer): integer;
begin
```

Мы не знаем наверняка, что задача имеет единственное решение, поэтому будем посчитывать варианты:

```
// число вариантов:
var nVar := 0;
```

Для каждой перестановки a мы находим общую годовую экономию и сравниваем её с заданной. Для этого пишем функцию GetSumma, которой передаём очередную перестановку чисел *1..4* и массив годовых экономий *eco*:

```
function GetSumma(a, eco: array of integer): boolean;
begin
```

```
    var n := a.Length - 2;
    var sum := 0;
```

Найти общую годовую экономию очень просто: перемножаем элементы из массивов a и eco с одинаковыми индексами и находим сумму этих произведений:

```
  for var i := 1 to n do
      sum += a[i] * eco[i];
```

Если полученная сумма совпадёт с указанной в задаче, то мы печатаем решение:

```
  if (sum = SUMMA) then
      PrintResult(a, eco);
   Result := sum = SUMMA;
end;
```

А функция *GetSumma* возвращает *true*, если решение найдено, или *false* в противном случае:

```
  repeat
      // нашли решение задачи:
      if (GetSumma(a, eco)) then
         nVar += 1;
```

Остальная часть кода функции *Permutation* не претерпела изменений, но возвращает она теперь не общее число перестановок, а число найденных решений задачи:

```
  Result := nVar;
end;
```

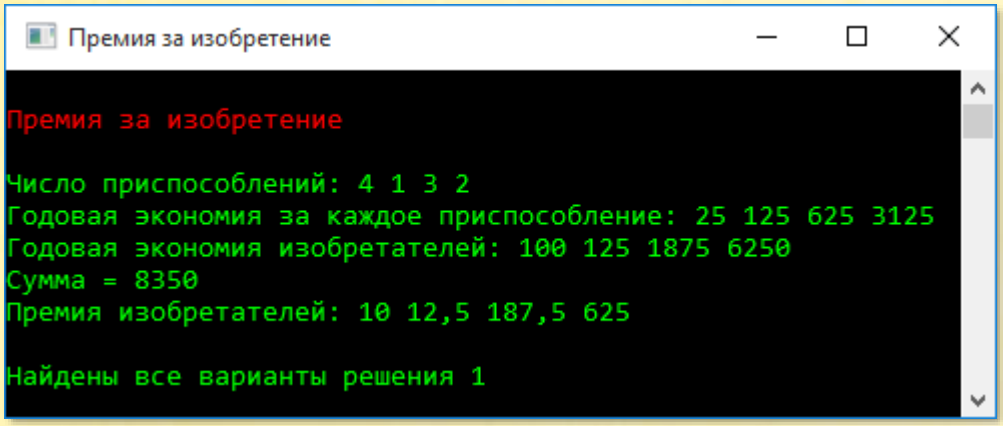
И последняя процедура проекта - PrintResult – обстоятельно информирует нас о решении задачи. Иначе говоря, распечатывает ту таблицу, которую мы заполняли вручную:

```
// ПЕЧАТАЕМ РЕШЕНИЕ ЗАДАЧИ
procedure PrintResult(a, eco: array of integer);
begin
   var n := a.Length - 2;

   Console.Write('Число приспособлений: ');
   for var i := 1 to n do
      Console.Write(a[i] + ' ');
   Console.WriteLine();

   Console.Write('Годовая экономия за каждое приспособление: ');

   for var i := 1 to n do
      Console.Write(eco[i] + ' ');

   Console.WriteLine();
   var sum := 0;
   Console.Write('Годовая экономия изобретателей: ');
   for var i := 1 to n do
   begin
      sum += a[i] * eco[i];
      Console.Write(a[i] * eco[i] + ' ');
   end;
   Console.WriteLine();

   Console.WriteLine('Сумма = ' + sum);

   Console.Write('Премия изобретателей: ');
   for var i := 1 to n do
      Console.Write(a[i] * eco[i] / 10.0 + ' ');
   Console.WriteLine();
end;
```

Итак, Григорий предложил 4 изобретения по 25 тыс. р., Андрей – 1 за 125, Виктор – 3 по 625 и Борис – 2 по 3125, что составляет как раз 8350 тыс. р.

Также вы видите, что каждый изобретатель получил 10% от суммы годовой экономии:

- Григорий - 10 тыс. р.
- Андрей – 12,5 тыс. р.
- Виктор – 187,5 тыс. р.
- Борис – 625 тыс. р.



А задача имеет *единственное* решение, напрасно мы сомневались…

Задания для самостоятельного решения

Поиграем в прятки

Удивительный мир чисел. Задача 8 (9)], страница 73

Все 10 цифр спрятались под буквами в каждом из пяти равенств:

1) Д • Г У В А = Б Ж Я И Е
2) Е • Д А И Г = Б • У В Ж Я
3) Е Ж • Г В А = Б У Я Д И
4) ЖИ • ДАГ = ЕУВБЯ
5) УА • ВБГ = ИЯ • ЖДА

Задача несложная, но трудоёмкая: чтобы решить её, придётся выписывать 10 вложенных циклов!

ЛОБ ТРИ САМ

Удивительный мир чисел. Задача 2, страница 70

Это трёхзначные числа, такие, что

ЛОБ +ТРИ = САМ

Расшифруйте сложение, обходясь без цифры нуль.

В любом возможном решении должна обнаружиться определенная закономерность в числе «САМ». Какая?

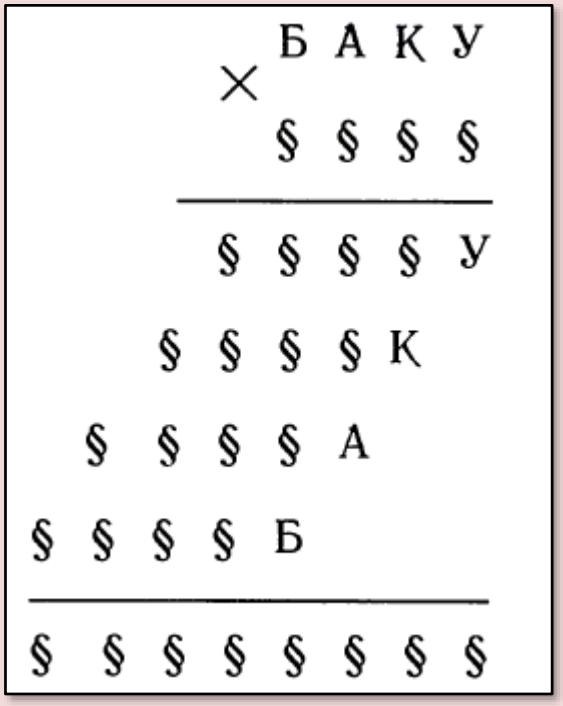
Семейство, спрятавшееся в «БАКУ»

Удивительный мир чисел. .Задача 5, страница 77

а) Расшифруйте произведение чисел **БАКУ** и **§§§§**, зная, что в каждом сомножителе цифры образуют (слева направо) строго убывающую последовательность.

Какое семейство цифр спряталось за буквами Б, А, К, У?

б) Расшифруйте в этом ребусе ещё и второй множитель (**§§§§**), если строго убывающую последовательность образуют цифры только первого сомножителя (**БАКУ**) и если в произведении вторая цифра слева - нуль.



Задача 11, страница 52

Удивительный мир чисел

Овчарка погналась за лисой, когда между ними было расстояние 99 м. Скачок лисы 1,1 м, скачок овчарки 2,2 м. Когда овчарка делает 19 скачков, лиса делает 29 скачков.

Сколько метров проскачут они, пока овчарка догонит лису?

Вы ошиблись в подсчёте

Удивительный мир чисел.. Задача 4 (2.2)], страница 50

Ученик покупает 18 карандашей, 6 тетрадей, 12 ластиков, 9 блокнотов и несколько тетрадей для рисования по 15 к. Девушка-продавец выписала чек на 1 р. 52 к. Взглянув на чек, мальчик сразу же сказал продавцу: «Вы ошиблись в подсчёте». Девушка пересчитала и исправила свою ошибку.

Как удалось пареньку так быстро обнаружить просчёт?

Ответ: Стоимость каждой покупки, а значит и общая сумма кратна трём, но 1 р. 52 к. на три не делится.

Задачи Анатолия Савина

Анатолий Павлович Савин известен как автор многих популярных книг по математике и ведущий раздела *«Квант» для младших школьников* в одноимённом журнале. Он придумал огромное число занимательных задач, которые интересно решать и вручную, и на компьютере. Мы, конечно, предпочтём второй способ, но вы можете параллельно решать задачи и на бумаге.

Самые интересные задачи Анатолия Савина собраны в книге Занимательные математические задачи:

Наименьшее число

> ***Найдите наименьшее число**, которое начинается с 1993 и делится на все числа от 1 до 9.*

Понятно, что задача, скорее, на сообразительность, чем переборная, но именно с помощью перебора её очень просто решить, что мы и сделаем в этом проекте.

Как обычно, главные события разворачиваются в процедуре Solve:

```
uses
   System;

// Савин 1-35

begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 1.35';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Наименьшее число');
   Console.ForegroundColor := ConsoleColor.Green;

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Будем считать, что переменная num равна началу числа, а endi – его концу:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;
   Console.WriteLine();

   var num := 1993;
   var endi := 0;
```

Если начало числа незыблемо и непоколебимо, то окончание числа изменяется от нуля и … пока мы не найдём решения задачи. Эта неопределённость диктует нам применение бесконечного цикла для поиска заданного числа:

```
while (true) do
begin
```

Конечную переменную endi мы не можем просто приставить в хвост к числу 1993 – сначала его следует умножить на 10, если переменная endi имеет значение меньше 10, на 100, если 10..99, и так далее.

Это процесс легко организовать в цикле while:

```
  var curnum := num * 10;
  var tmp := endi;
  while(tmp div 10 > 0) do
  begin
     curnum *= 10;
     tmp := tmp div 10;
end;
```

В итоге наших объединяющих усилий мы получим в переменной curnum число 1993end:

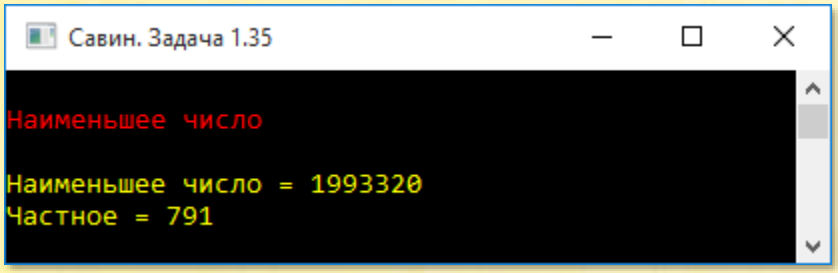
```
curnum += endi;
endi += 1;
```

Чтобы проверить его делимость на ряд чисел 1..9, достаточно привлечь к испытаниям только числа 5, 7, 8 и 9, остальные удовлетворятся автоматически:

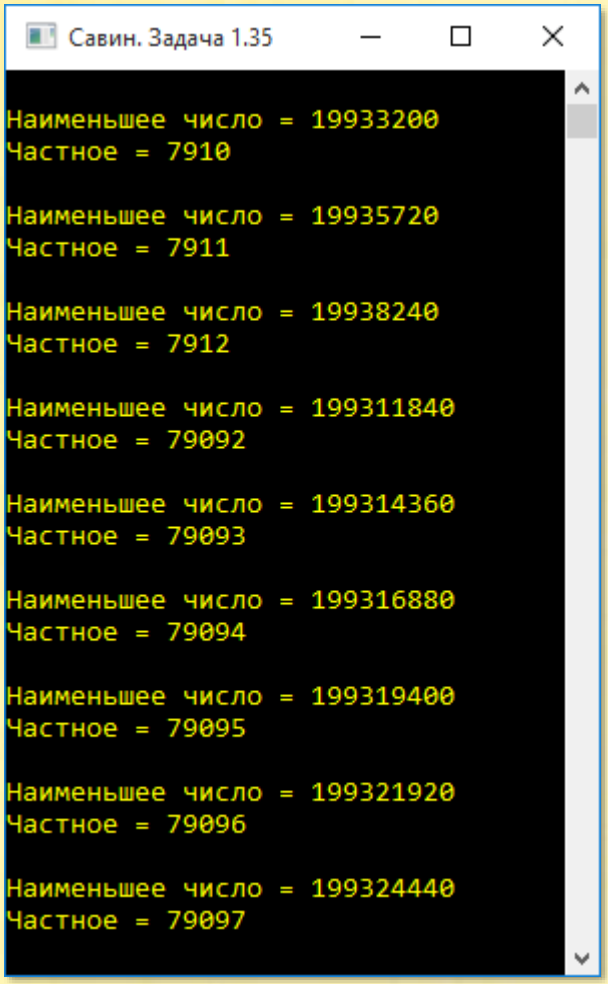
```
if ((curnum mod 5 = 0) and
(curnum mod 7 = 0) and
(curnum mod 8 = 0) and
(curnum mod 9 = 0)) then
begin
```

Как только мы обнаружим первое число, удовлетворяющее условию задачи, мы печатаем ответ на экране и выходим из бесконечного цикла *while*:

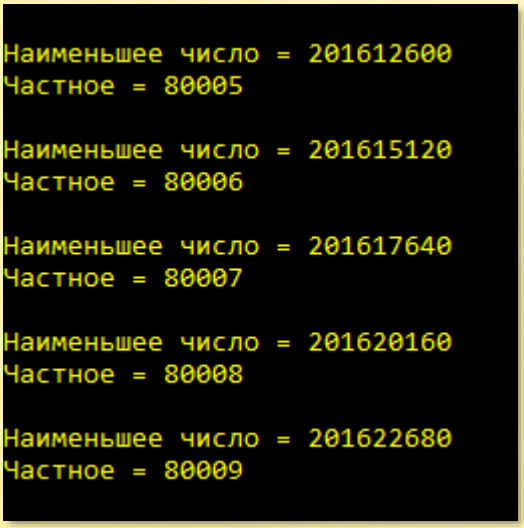
```
            Console.WriteLine('Наименьшее число = ' + curnum);
            Console.WriteLine('Частное = ' + curnum div 5 div 7 div 8
                                           div 9);
            Console.WriteLine();
            exit;
        end
    end;
end;
```

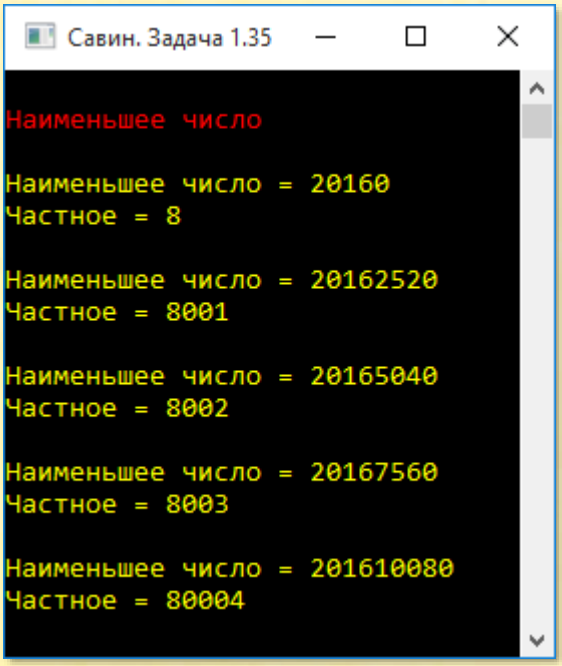


При известном любопытстве можно найти ещё несколько решений этой задачи, если не требовать, чтобы числа были наименьшими из возможных:



Чем хороши компьютерные решения, так это тем, что с их помощью можно запросто перерешать множество подобных задач. Например, исправив значение единственной переменной **num**, мы тут же получим ответ на задачу для *2016* года:





Когда он родился?

Моему племяннику в x^2 году исполнится x лет.

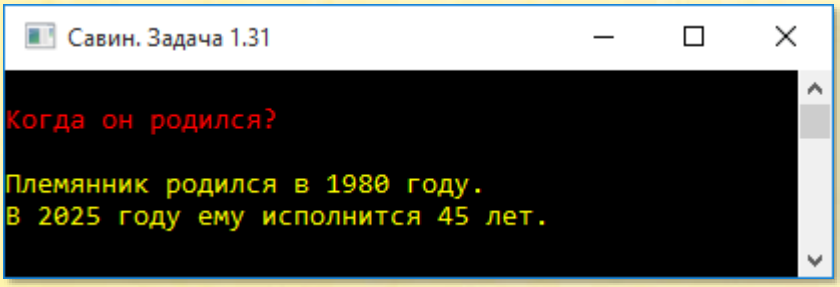
В каком году он родился?

Понятно и очевидно, что нужно перебирать возраст племянника до тех пор, пока квадрат его возраста не превысит текущего года:

```
uses
   System;

// Савин 1-31


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var x := 0;
   while (true) do
   begin
      if (x * x > 2014) then
      begin
         Console.WriteLine('Племянник родился в {0} году.',
                           x * x - x);
         Console.WriteLine('В {0} году ему исполнится {1} лет.',
                           x * x, x);
         Console.WriteLine();
```

```
            break;
        end;
        x += 1;
    end;
end;


begin
    // заголовок окна:
    Console.Title := 'Савин. Задача 1.31';
    Console.WriteLine('');
    Console.ForegroundColor := ConsoleColor.Red;
    Console.WriteLine('Когда он родился?');
    Console.ForegroundColor := ConsoleColor.Green;
    Console.WriteLine();

    Solve();

    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```

Ответ представлен на рисунке:



Дед и внуки

Дедушка с тремя внуками вышел прогуляться в парк. Встретившийся им дедушкин знакомый спросил, сколько каждому из них лет.

Ваня сказал: «Я моложе Пети и мне больше трёх лет».
Петя произнёс: «Я моложе Саши на четыре года».
А Саша сказал: «Нам вместе в четыре раза меньше лет, чем дедушке, а вместе с дедушкой нам ровно 100 лет».

Сколько лет каждому из внуков?

Решаем задачу методом грубой силы:

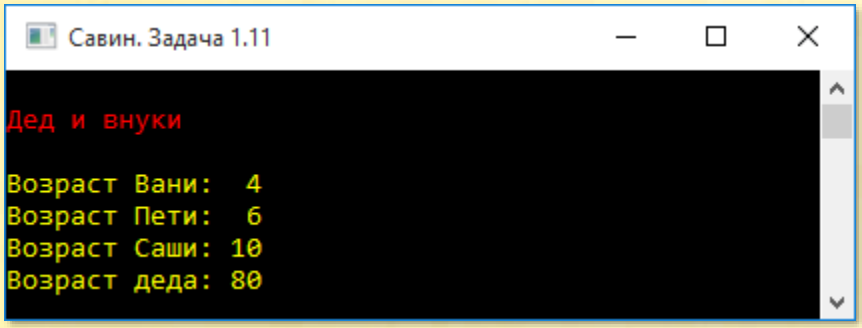
```
uses
   System;

// Савин 1-11

begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 1.11';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Дед и внуки');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();
```

```
   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Совершенно очевидно, что каждый из внуков младше дедушки, то есть ему заведомо меньше 100 лет. Этого немудрёного умозаключения вполне достаточно для того чтобы решить задачу:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   for var v := 4 to 100 - 1 do
      for var p := v + 1 to 100 - 1 do
      begin
         var s := p + 4;
         var d := (v + s + p) * 4;
         if ((v + s + p + d) = 100) then
         begin
            Console.WriteLine('Возраст Вани:  ' + v);
            Console.WriteLine('Возраст Пети:  ' + p);
            Console.WriteLine('Возраст Саши: ' + s);
            Console.WriteLine('Возраст деда: ' + (v + s + p) * 4);
            Console.WriteLine();
            break;
         end
      end
end;
```

А вот и ответ:



Делимость на 92

В клетки прямоугольника 3 x 4 впишите цифры так, чтобы в каждом горизонтальном ряду стояло четырёхзначное число, делящееся на 92, а в каждом вертикальном ряду – трёхзначное число, тоже делящееся на 92.

Если вы заглянете в ответ, то обнаружите, что прямоугольная таблица имеет размеры не 3 x 4 клетки, а 4 x 3:

3.44. Требуемое заполнение таблицы указано на рисунке.

2	3	5	7
6	3	7	5
3	4	6	8

Но самое интересное ещё впереди! Легко проверить, что ни одно из чисел в таблице на 92 не делится!

Давайте решать задачу самостоятельно...

Метод грубой силы тут не проходит, поскольку в 12 клеток можно расставить 10 цифр слишком большим числом способов.

Зато трёхзначных чисел, кратных 92, всего 9 штук. В условии задачи не запрещается повторять их в таблице, поэтому их можно расставить всего 9 x 9 x 9 x 9 = 6561 способом. Для компьютера - это сущая безделица.

Найти все трёхзначные числа, делящиеся на 92 без остатка, ещё проще, но для формирования четырёхзначных чисел, нам потребуются **цифры** трёхзначных чисел, и функция **GetDigits**, которая возвратит нам цифры заданного числа в виде целочисленного массива:

```
// НАХОДИМ ЧИСЛО ЦИФР В ЗАДАННОМ ЧИСЛЕ
function NumDigit(num: int64): integer;
begin
   num := System.Math.Abs(num);
   var nd := 0;
   while (num > 0) do
   begin
      nd += 1;
      num := num div 10;
   end;
   Result := System.Math.Max(1, nd);
end;

// ВОЗВРАЩАЕМ МАССИВ ЦИФР ЗАДАННОГО ЧИСЛА
function GetDigits(num: int64): array of integer;
begin
   num := System.Math.Abs(num);
   // число цифр:
   var n := NumDigit(num);
   // массив цифр:
   var res := new integer[n];
   var id := 0;
   while (num > 0) do
   begin
      id += 1;
      res[n - id] := integer(num mod 10);
      num := num div 10;
   end;
```

```
   Result := res;
end;
```

Решение задачи мы традиционно начинаем в **главном блоке** программы, а непосредственно решаем задачу в процедуре *Solve*:

```
uses
   System;

// Савин 3-44

begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 3.44';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Делимость на 92');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

В процедуре **Solve**, в цикле *for* мы находим все трёхзначные числа, кратные 92, получаем от функции *GetDigits* массив их цифр и сохраняем его в списке *lst*:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
type
   aoi = array of integer;
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   // трёхзначные числа, кратные 92:
   var lst := new List<aoi>();

   for var i := 100 div 92 + 1 to 999 div 92 do
   begin
```

```
      // число:
      var n := 92 * i;
      // массив цифр:
      var m := GetDigits(n);
      lst.Add(m);
   end;
```

По условию задачи, мы должны вписать 4 трёхзначных числа в таблицу вертикальными рядами, а затем проверить получившиеся в горизонталях четырёхзначные числа на делимость:

```
   // четырёхзначные числа:
   var num4 := new integer[3];
   foreach var a1 in lst do
      foreach var a2 in lst do
         foreach var a3 in lst do
            foreach var a4 in lst do
            begin
               var n4 := 0;
               var flg := true;
               for var i := 0 to 3 - 1 do
               begin
                    //четырёхзначное число:
                  n4 := a1[i] * 1000 +
                        a2[i] * 100 +
                        a3[i] * 10 +
                        a4[i];
```

Если очередное четырёхзначное число не делится на 92 без остатка, то остальные и проверять не нужно:

```
if (n4 mod 92 <> 0) then
begin
   flg := false;
   break;
end;
```

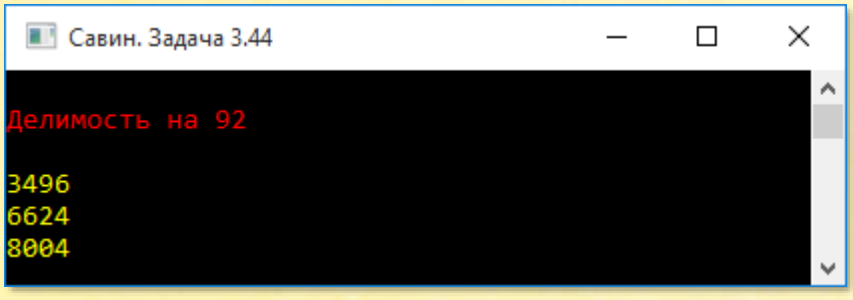
Если же четырёхзначное число удовлетворяет условию задачи, то мы запоминаем его в массиве num4:

```
  num4[i] := n4;
end;
```

Если все 3 четырёхзначных числа кратны 92, то переменная flg останется верной. Это значит, что все четырёхзначные числа в таблице выдержали проверку на делимость (трёхзначные заведомо кратны числу 92), и мы можем напечатать ответ на задачу:

```
            if (flg) then
            begin
               foreach var n in num4 do
                  Console.WriteLine(n);
               Console.WriteLine();
            end
         end
end;
```

После ошибочного ответа в книге мы могли бы усомниться в самом условии задачи. Но нет – рисунок ясно свидетельствует, что автор составил задачу корректно: она не только решается, но и имеет *единственное* решение.



А вот кто тот стрелочник, который напечатал неправильный ответ, мы не узнаем никогда...

Рыбный день

Гавиал, бегемот, пеликан и кашалот съели в общей сложности 37 рыб, причём кашалот съел во столько же раз больше пеликана, во сколько пеликан съел больше гавиала.

Сколько съел каждый?

Хитрость задачи заключается в том, что бегемоты рыбу не едят. Стало быть, у нас остаётся только трое едоков: *кашалот, гавиал* и *пеликан*.

Обозначим их первыми буквами: k, g, p.

Тогда, по условию задачи, получаем:

$$\frac{k}{p} = \frac{p}{g}$$

Или:

$$k*g = p*p$$

Будем полагать, что каждый из них съел хотя бы одну рыбку, хотя последнее равенство выполняется и при нулевых значениях.

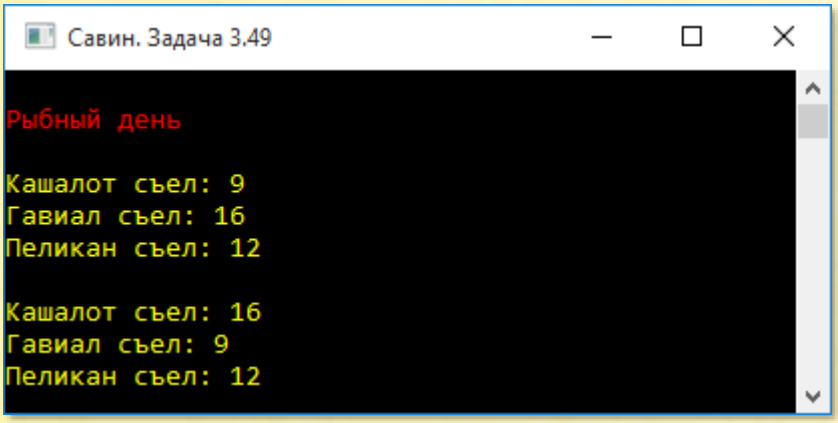
Таким образом, задача решается простым **перебором**. Действительно, кашалот мог съесть от 1 до 35 рыб, гавиал - от 1 до 35 за вычетом рыб, съеденных кашалотом. А на долю пеликана приходятся все оставшиеся рыбки.

```
uses
   System;

// Савин 3-49


begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 3.49';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Рыбный день');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Для перебора вариантов достаточно двух вложенных циклов *for* и одной простой проверки:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   for var k := 1 to 35 do
      for var g := 1 to 35 - k do
      begin
         // число рыб пеликана:
         var p := 37 - k - g;
         if (k * g = p * p) then
         begin
            Console.WriteLine('Кашалот съел: ' + k);
            Console.WriteLine('Гавиал съел: ' + g);
            Console.WriteLine('Пеликан съел: ' + p);
            Console.WriteLine();
         end
      end
end;
```

Мы получили 2 ответа на задачу, хотя в книге указан только второй:



Очевидно, слово «больше» следует понимать в бытовом, а не в математическом смысле. Тогда кашалот съел в 16/12 раз больше пеликана, а пеликан, в свою очередь, в 12/9 раз больше гавиала. Наш первый ответ также правильный: кашалот съел в 12/16 раз больше пеликана, а пеликан, в свою очередь, в 9/12 раз больше гавиала. Но этот ответ слишком абстрактен для рядового поедателя рыб.

Удивительное число

*Из 10 различных цифр **составьте десятизначное число**, такое, что число из его первых 2-х цифр делится на 2, из 3-х первых цифр – на 3 и т.д. до того, что само число делится на 10.*

Задачу вполне можно решить полным перебором, но в данном случае не стоит лишний раз напрягать компьютер – достаточно немного напрячь мозги, чтобы существенно сократить перебор.

Например, из признаков делимости ясно следует, что последняя цифра числа – 0, а пятая – 5.

А также: все цифры на чётных местах – чётные, а это значит, что все цифры на нечётных местах – нечётные.

Теперь задачу вполне можно решить ручным перебором, как это сделал автор, но это занятие длинное и скучное, поэтому мы по свежим следам быстро напишем программу, которая вмиг найдёт решение задачи.

```
uses
   System;

// Савин 3-50

begin
```

```
   // заголовок окна:
   Console.Title := 'Савин. Задача 3.50';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Удивительное число');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

В процедуре Solve мы изящно используем признаки делимости чисел, а весь остальной код слишком прост, чтобы его комментировать:

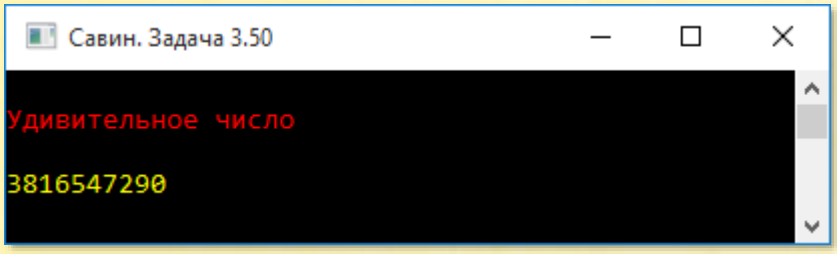
```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var d10 := 0;
   var d5 := 5;

   var d1 := 1;
   while (d1 <= 9) do
   begin
      if (d1 = d5) then begin
         d1 += 2;
         continue;
      end;
      var d2 := 2;
      while (d2 <= 8) do
      begin
         var d3 := 1;
         while (d3 <= 9) do
         begin
            if ((d3 = d1) or (d3 = d5)) then begin
               d3 += 2;
               continue;
            end;
```

```
if ((d1 + d2 + d3) mod 3 <> 0) then begin
   d3 += 2;
   continue;
end;
var d4 := 2;
while (d4 <= 8) do
begin
   if (d4 = d2) then begin
      d4 += 2;
      continue;
   end;
   if ((d3 * 10 + d4) mod 4 <> 0) then begin
      d4 += 2;
      continue;
   end;
   var d6 := 2;
   while (d6 <= 8) do
   begin
      if ((d6 = d2) or (d6 = d4)) then
      begin
         d6 += 2;
         continue;
      end;

      if ((d1 + d2 + d3 + d4 + d5 + d6) mod 3 <> 0) then
      begin
         d6 += 2;
         continue;
      end;
      var d7 := 1;
      while (d7 <= 9) do
      begin
         if ((d7 = d1) or (d7 = d3) or (d7 = d5)) then
         begin
            d7 += 2;
            continue;
         end;

         if ((d1 * 1000000 + d2 * 100000 + d3 * 10000 +
              d4 * 1000 + d5 * 100 + d6 * 10 + d7)
              mod 7 <> 0) then
         begin
            d7 += 2;
            continue;
         end;
```

```
                var d8 := 2;
                while (d8 <= 8) do
                begin
                   if ((d8 = d2) or (d8 = d4) or (d8 = d6))
                      then begin
                      d8 += 2;
                      continue;
                   end;
                   if ((d6 * 100 + d7 * 10 + d8) mod 8 <> 0)
                      then begin
                      d8 += 2;
                      continue;
                   end;
                   var d9 := 1;
                   while (d9 <= 9) do
                   begin
                      if ((d9 = d1) or (d9 = d3) or (d9 = d5)
                         or (d9 = d7)) then begin
                         d9 += 2;
                         continue;
                      end;
                      if ((d1 + d2 + d3 + d4 + d5 + d6 + d7 +
                           d8 + d9) mod 9 <> 0) then begin
                         d9 += 2;
                         continue;
                      end;
                      Console.WriteLine('' + d1 + d2 + d3 +
                                        d4 + d5 +
                                        d6 + d7 + d8 + d9 + d10);
                      Console.WriteLine();
                      d9 += 2;
                   end;
                   d8 += 2;
                end;
                d7 += 2;
             end;
             d6 += 2;
          end;
          d4 += 2;
       end;
       d3 += 2;
    end;
    d2 += 2;
end;
d1 += 2;
```

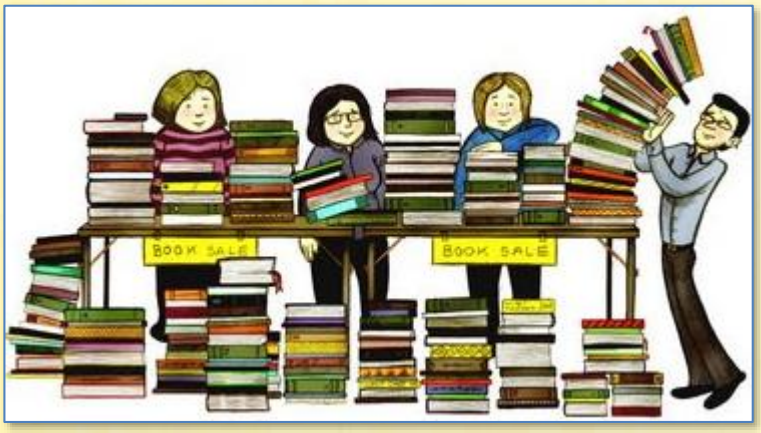
```
    end
end;
```

Довольно странно, но из огромного множества чисел, только одно-единственное удовлетворяет всем условиям задачи:



Как говорится, скрупулёзно подмечено!

Продажа книг



В три магазина привезли 1990 книг. В первые три дня первый магазин продал соответственно 1/37, 1/11, 1/2 полученных книг, второй магазин 1/57, 1/9 и 1/3 полученных книг, третий магазин 1/25, 1/30, 1/10 полученных книг.

Сколько книг получил каждый магазин?

Так как книги пока ещё продают штуками, а не россыпью, то все магазины каждый день продавали **целое** число книг. Это значит, что число книг, полученных каждым магазином, должно делиться нацело:

- на 37, 11, 2 – для **первого** магазина
- на 57, 9, 3 – для **второго** магазина
- на 25, 30,10 – для **третьего** магазина

Минимальное число полученных книг равно наименьшему общему кратному этих чисел (*НОК*).

Теперь мы можем взяться за непосредственное решение задачи, используя функцию **NOK**:

```
// БЫСТРЫЙ АЛГОРИТМ ЕВКЛИДА
function SpeedEuklid(n1, n2: int64): int64;
begin
   while (n2 > 0) do
   begin
      var n := n1 mod n2;
      n1 := n2;
      n2 := n;
   end;
   Result := n1;
end;

// ВЫЧИСЛЯЕМ НОК
function NOK(number1, number2: int64): int64;
begin
   if (number1 * number2 = 0) then
      Result := 0
   else
      Result := number1 * number2 div SpeedEuklid(number1, number2);
end;



uses
   System;

// Савин 3-52
```

```
begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 3.52';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Продажа книг');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
const
   BOOKS = 1990;
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var nok1 := NOK(NOK(37, 11), 2);
   var nok2 := NOK(NOK(57, 9), 3);
   var nok3 := NOK(NOK(25, 30), 10);
   Console.WriteLine('НОК1 = ' + nok1 + NewLine + 'НОК2 = ' + nok2 +
                     NewLine + 'НОК3 = ' + nok3);
   Console.WriteLine();
```

На рисунке вы видите, что первый магазин получил как минимум 814 книг, второй – 171 и третий – 150:

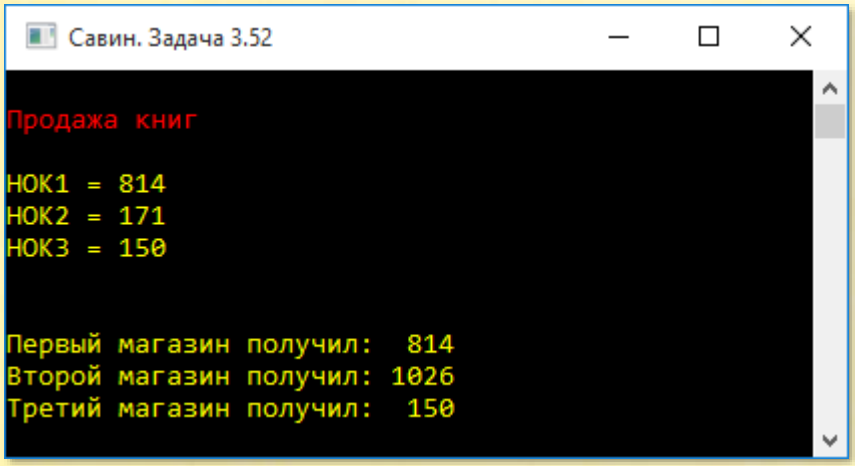
Легко посчитать, что сумма этих чисел меньше общего числа полученных книг, поэтому все или некоторые из полученных нами чисел следует умножить на какие-то, пока нам неизвестные «коэффициенты» так, чтобы получить 1990 книг.

Множитель для минимального числа проданных **первым** магазином книг находится в диапазоне *1.. BOOKS / nok1*, **вторым** магазином *1.. BOOKS / nok2* и **третьим** магазином - *1.. BOOKS / nok3*.

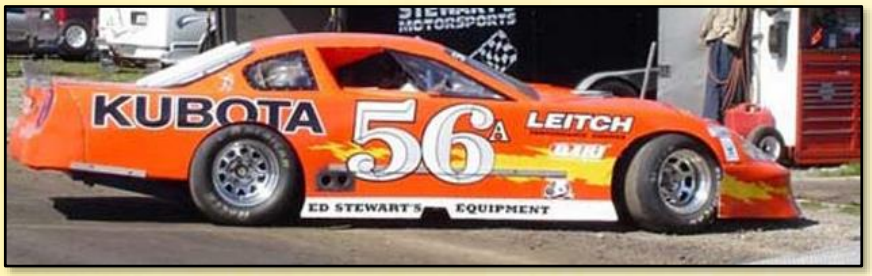
Так как диапазоны перебора значений множителей очень невелики, то мы быстро пишем 3 вложенных цикла:

```
  for var m1 := 1 to BOOKS div nok1 do
      for var m2 := 1 to BOOKS div nok2 do
         for var m3 := 1 to BOOKS div nok3 do
            if (m1 * nok1 + m2 * nok2 + m3 * nok3 = BOOKS) then
            begin
               Console.WriteLine(NewLine + 'Первый магазин получил:  ' +
                                 m1 * nok1 +
                                 NewLine + 'Второй магазин получил: ' +
                                 m2 * nok2 +
                                 NewLine + 'Третий магазин получил:  ' +
                                 m3 * nok3);
               Console.WriteLine();
            end
end;
```

И печатаем **ответ** на экране:



Кругом 56



Найдите наименьшее натуральное число, которое оканчивается на 56, делится на 56 и имеет сумму цифр, равную 56.

А существует ли аналогичное число, в котором вместо 56 фигурирует 11?

Так как при решении задачи нам необходимо знать сумму цифр числа, то мы напишем вспомогательную функцию Summa:

```
// НАХОДИМ СУММУ ЦИФР ЗАДАННОГО ЧИСЛА
function Summa(num: int64): integer;
begin
   var sum: int64 := 0;
   while (num > 0) do
   begin
      sum += num mod 10;
      num := num div 10;
   end;
   Result := integer(sum);
end;
```

Чтобы сделать программу более универсальной, введём константу NUM, значение которой легко исправить:

```
uses
   System;
```

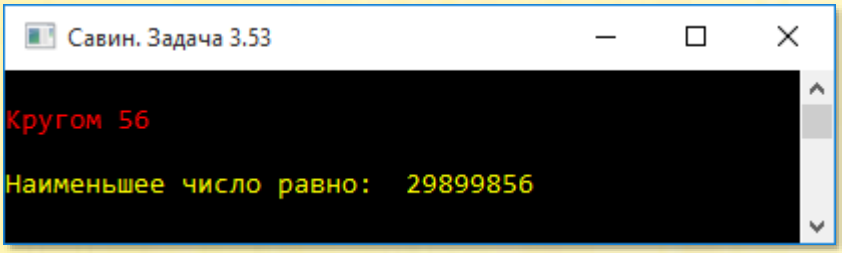
```
// Савин 3-53

begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 3.53';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Кругом 56');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
const
   NUM = 56;
begin
   Console.ForegroundColor := ConsoleColor.Yellow;
```

Первое число, которое заканчивается на 56, это само число 56, следующее число на 100 больше, следующее за ним – ещё на 100 больше. Следовательно, нам нужно перебирать такие числа в бесконечном цикле ***while*** до тех пор, пока не выполнятся оба условия задачи:

```
   var i := NUM - 100;
   while (true) do
   begin
      i += 100;
      if (i mod NUM <> 0) then
         continue;
      if (Summa(i) <> NUM) then
         continue;
```

```
        Console.WriteLine('Наименьшее число равно:  ' + i);
        Console.WriteLine();
        break;
    end
end;
```

Наименьшее число совсем немаленькое, но мы находим его мгновенно:



Уберите оператор *break* – и вы получите множество чисел, удовлетворяющих условию задачи.

Вторая часть задачи в ответе не упоминается. Вероятно, это шутка автора задачи…

> Если заданное число заканчивается на 11, то одна из цифр, стоящих на чётных и нечётных местах, равна 1. В данном случае, по признаку делимости на 11, суммы цифр искомого числа, стоящих на чётных и нечётных местах, должны быть равны. Но 11 нельзя поделить на 2 поровну. Следовательно, не существует такого числа, о котором говорится в задаче.

Факториалы

Факториалом числа (обозначается !) называется произведение всех натуральных чисел от 1 до n .

Например, 3! = 1 x 2 x 3 = 6;
5! = 1 x 2 x 3 x 4 x 5 = 120.

Найдите два семизначных числа, такие, что их сумма является факториалом некоторого числа, их разность и сумма цифр одного из этих чисел также являются факториалами.

В этом проекте мы предварительно напишем функцию FactorialArr, которая возвращает массив факториалов чисел от 1 до *max*:

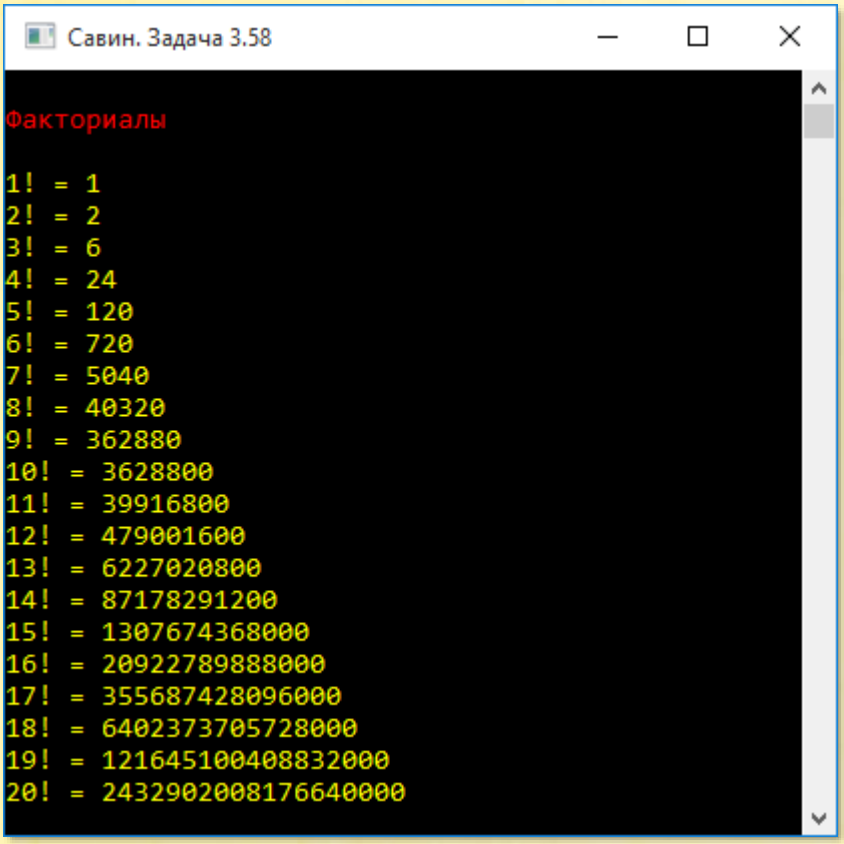
```
// ВЫЧИСЛЯЕМ МАССИВ ФАКТОРИАЛОВ 0..max
function FactorialArr(max: integer): array of int64;
begin
   // массив для хранения факториалв:
   var factorial := new int64[max + 1];
   // особый случай: 0! = 1
   factorial[0] := 1;

   for var i := 1 to max do
      factorial[i] := factorial[i - 1] * i;
```

```
   Result := factorial;
end;
```

Давайте поинтересуемся факториалами первых 20 чисел:

```
uses
   System;

// Савин 3-58

begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 3.58';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Факториалы');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var facts := FactorialArr(20);
   for var i := 1 to facts.Length - 1 do
      Console.WriteLine(i + '! = ' + facts[i].ToString());
```

Рисунок показывает, что факториалы стремительно увеличиваются!



Мы легко найдём единственный 7-значный факториал 3628800 и также единственный 8-значный факториал 39916800. Первый факториал может быть суммой двух семизначных чисел, а вот второй – нет. Действительно, сумма двух семизначных чисел не больше, чем 9 999 999 + 9 999 999 = 19999998, а это меньше 8-значного факториала.

Итак, сумма двух семизначных чисел равна 10! = 3628800.

Обозначим большее из двух чисел буквой a, а меньшее - буквой b. Тогда минимальное значение числа *b = 10000000*. Минимальное значение числа a = *3628800/2*, а максимальное – *3628800 – b*.

Все возможные значения числа a мы перебираем в цикле *for*:

```
for var a := 3628800 div 2 to 3628800 - 1000000 do
begin
```

Находим значение числа b:

```
var b := 3628800 - a;
```

Их разность должна быть факториалом, а все необходимые факториалы мы поместили в массив facts, поэтому нам достаточно проверить, имеется ли нужный факториал в этом массиве:

```
if (not (facts.Contains(int64(a - b)))) then
   continue;
```

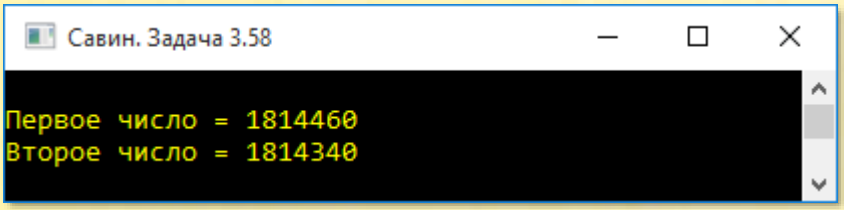
Аналогично мы проверяем и третье условие задачи, что сумма цифр одного из искомых чисел также является факториалом:

```
if (not facts.Contains(int64(Summa(a))) and
not facts.Contains(int64(Summa(b)))) then
   continue;
```

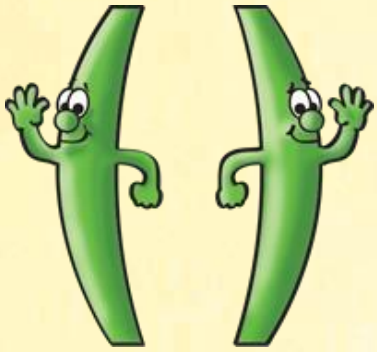
Все пары чисел, прошедшие проверки, мы печатаем на экране:

```
      Console.WriteLine();
      Console.WriteLine('Первое число = ' + a);
      Console.WriteLine('Второе число = ' + b);
      Console.WriteLine();
   end
end;
```

На рисунке вы видите, что такая пара чисел – *единственная*:



Расставьте скобки



Расставьте скобки в левой части равенства так, чтобы оно стало верным.

1 : 2 : 3 : 4 : 5 : 6 : 7 : 8 : 9 : 10 = 7.

Пусть в левой части равенства стоят только **два** числа:

1 : 2

В этом случае скобки расставлять не нужно – мы всегда получим дробь:

$$\frac{1}{2}$$

Добавим **третье** число:

1 : 2 : 3

Расставим скобки всеми возможными способами:

(1 : 2) : 3
1 : (2 : 3)

Запишем эти выражения в виде дробей:

$$\frac{1}{2} : 3 = \frac{1}{2 * 3}$$

$$\frac{1}{2:3} = \frac{1 * 3}{2}$$

Если вы добавите **четвёртое** число, то получите такие дроби:

$$\frac{1}{2 * 3 * 4}$$

$$\frac{1 * 4}{2 * 3}$$

$$\frac{1 * 3}{2 * 4}$$

$$\frac{1 * 3 * 4}{2}$$

При добавлении следующих чисел вы всегда будете получать дробь, в числителе которой стоит **1**, а в знаменателе – **2**. Все остальные числа из множества *{1..10}* распределяются между числителем и знаменателем всеми возможными способами.

Пусть произведение всех чисел в числителе, за исключением 1, равно **a**, а произведение всех чисел в знаменателе, за исключением 2, равно **b**. Тогда

$$\frac{1 * a}{2 * b} = 7$$

Или

$$\frac{1 * a}{2 * b * 7} = 1$$

Так как изо всех чисел множества *{1..10}* только семёрка делится нацело на 7, то она должна стоять в числителе:

$$\frac{1 * 7}{2 * 7} \cdot \frac{a}{b} = 1$$

Откуда: $a = 2b$.

Мы уже нашли места для чисел 1, 2 и 7, поэтому нам осталось распределить числа 3, 4, 5, 6, 8, 9, 10 между числителем и знаменателем.

Причём $2b*b = 3 * 4 * 5 * 6 * 8 * 9 * 10$ →

$b^2 = 3 * 2 * 5 * 6 * 8 * 9 * 10 = 1440 * 90 = 144 * 900$ → $b = 12 * 30 = 360$

Итак,

$b = 360$
$a = 2b = 720$

Чтобы получить произведение 360, необходимо не менее трёх чисел, так как 9 * 10 = 90. И не более четырёх, так как 2 * 3 * 4 * 5 = 360.

Следовательно, из множества *{ 3, 4, 5, 6, 8, 9, 10 }* нам нужно выбрать 3 или 4 числа, произведение которых равняется 360.

Тогда на долю числителя останется 4 или 3 числа, произведение которых равняется 720.

Теперь перебор стал простым и понятным. Мы берём 3 разных числа из множества *{ 3, 4, 5, 6, 8, 9, 10 }* и находим их произведение. Если оно равно 360, значит, это тройка чисел знаменателя, а все остальные числа из множества *{ 3, 4, 5, 6, 8, 9, 10 }* принадлежат числителю.

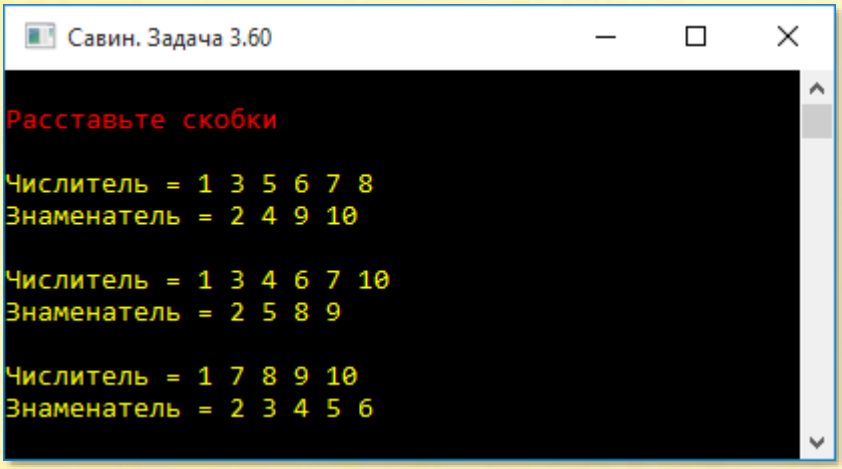
Если же произведение равно 720, то это тройка чисел числителя, а все остальные числа из множества *{ 3, 4, 5, 6, 8, 9, 10 }* принадлежат знаменателю.

Дальше комментировать программу не обязательно:

```
uses
   System;

// Савин 3-60


// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   var numbers := new List<integer>(Seq(3, 4, 5, 6, 8, 9, 10 ));

   for var n1 := 0 to numbers.Count - 2 - 1 do
      for var n2 := n1 + 1 to numbers.Count - 1 - 1 do
         for var n3 := n2 + 1 to numbers.Count - 1 do
         begin
            // произведение:
            var product := numbers[n1] * numbers[n2] * numbers[n3];
            if (product = 360) then
            begin
               var z := new List<integer>(Seq(2, numbers[n1],
               numbers[n2],
               numbers[n3] ));
               var ch := new List<integer>(Seq(1, 7 ));
               foreach var i in numbers do
                  if (not z.Contains(i)) then
                     ch.Add(i);
               ch.Sort();
               Console.Write('Числитель = ');
               foreach var i in ch do
                  Console.Write(i + ' ');
               Console.WriteLine();

               Console.Write('Знаменатель = ');
               foreach var i in z do
                  Console.Write(i + ' ');
               Console.WriteLine();
               Console.WriteLine();
            end;
            if (product = 720) then
            begin
```

```
                var z := new List<integer>(Seq(2));
                var ch := new List<integer>(Seq(1, 7,
                numbers[n1],
                numbers[n2],
                numbers[n3] ));
                foreach var i in numbers do
                   if (not ch.Contains(i)) then
                      z.Add(i);
                z.Sort();
                Console.Write('Числитель = ');
                foreach var i in ch do
                   Console.Write(i + ' ');
                Console.WriteLine();

                Console.Write('Знаменатель = ');
                foreach var i in z do
                   Console.Write(i + ' ');
                Console.WriteLine();
                Console.WriteLine();
             end
          end;
   Console.WriteLine();
end;

begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 3.60';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Расставьте скобки');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

На рисунке вы видите, как следует разбить на 2 группы заданные числа.



А расставить скобки теперь уже совсем просто!

Награда калифа. Савин. Задача 3.66

- Кошелёк, кошелёк, какой кошелёк?

Калиф Гарун-аль-Рашид одарил троих придворных астрологов десятью кошельками.

Мудрецы, сев подсчитывать доход, обнаружили, что один кошелёк пуст, во втором лежит одна таньга, в третьем – две и так далее, до десятого, в котором оказалось девять таньга.

Гуссейн Гуслия взял себе два кошелька. Абдурахман ибн Хоттаб и его брат Омар Юсуф разделили оставшиеся кошельки так, что более заслуженный и умудрённый годами Абдурахман получил бо́льшую сумму денег.

По дороге домой на Омара Юсуфа напали разбойники и отняли 4 кошелька, так что от подарка калифа у него осталось лишь 10 таньга.

Какие кошельки достались Гуссейну Гуслия?

Омар Юсуф получил 4 кошелька и ещё 10 таньга. Но 10 таньга могли лежать как минимум в 2 кошельках. Итого Омар Юсуф получил не меньше 6 кошельков.

Ещё 2 кошельками был одарён **Гуссейн Гуслия**. Тогда на долю **Абдурахмана** остаётся не более 2 кошельков. Если бы он получил 1 кошелёк, то в нём оказалось бы не более 9 таньга, а это меньше, чем у Омара Юсуфа.

Следовательно, **Омар Юсуф** получил 6 кошельков, а **Гуссейн Гуслия** и **Абдурахман** – по 2.

В 4 кошельках **Омар Юсуфа**, отобранных разбойниками, было не менее 0 + 1 + 2 + 3 = 6 таньга. Вместе с оставшимися деньгами это не менее 16 таньга. Значит, **Абдурахман** получил не менее 17 таньга, но тогда в его кошельках должны были лежать 8 и 9 таньга.

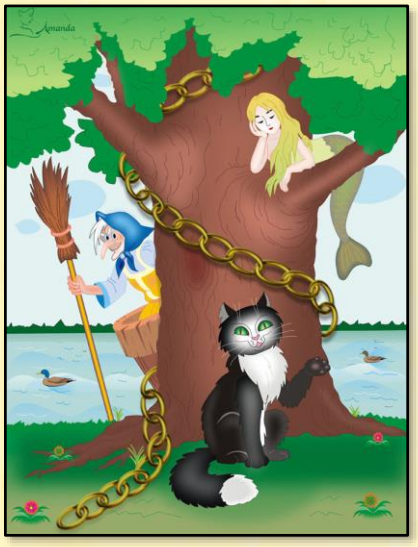
Имеем:

Кошельки с 0 1 2 3 таньга получил **Омар Юсуф**.
Кошельки с 8 и 9 таньга получил **Абдурахман**.
В двух кошельках **Омар Юсуфа** было ещё 10 таньга. Из оставшихся кошельков с 4 5 6 7 таньга только кошельки с 4 и 6 таньга могли дать такую сумму.

А Гуссейну Гуслия остаются кошельки с 5 и 7 таньга.

Задача (раз)решилась сама собой, так что компьютер нам не понадобится.

Другая последовательность



Если выписать цифры от 1 до 9 подряд 1,2, . . . , 9, то сумма пар соседних чисел: 3, 5, 7 … 17 увеличивается на 2.

Расположите эти цифры в такой последовательности, чтобы суммы соседних чисел увеличивались на 1.

Обозначим однозначные числа буквами и расставим их по условию задачи:

a b c d t f g h i

Пусть сумма первых двух чисел равна s1, тогда:

s2 = s1 + 1
s3 = s1 + 2
s4 = s1 + 3
s5 = s1 + 4
s6 = s1 + 5
s7 = s1 + 6
s8 = s1 + 7

А сумма всех сумм равна:

s1 * 8 + 28

С другой стороны:

s1 = a + b
s2 = b + c
s3 = c + d
s4 = d + e
s5 = e + f
s6 = f + g
s7 = g + h
s8 = h + i

А сумма всех сумм:

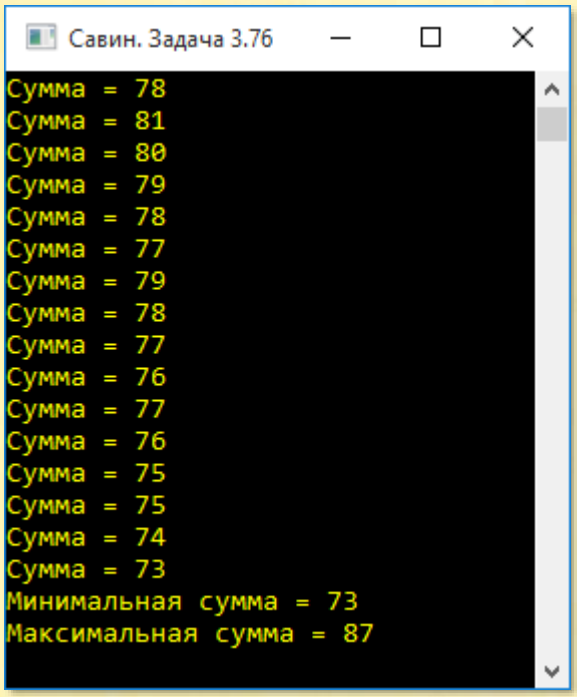
a + i + (b + c + d + e + f + g + h)*2

Минимальное значение суммы сумм мы получим, если в скобках стоят числа 1..7, а максимальное – если 3..9.

Вы можете проверить это утверждение:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   // минимальная сумма:
   var min := 1000;
   // максимальная сумма:
   var max := 0;
   // список чисел:
   var numbers := new List<int>(Seq(1, 2, 3, 4, 5, 6, 7, 8, 9 ));

   for var n1 := 1 to 8 do
      for var n2 := n1 + 1 to 9 do
      begin
         // сумма:
         var sum := n1 + n2;
         foreach var n in numbers do
            if ((n <> n1) and (n <> n2)) then
```

```
                sum += (n + n);
            if (sum < min) then
               min := sum;
            if (sum > max) then
               max := sum;
            Console.WriteLine('Сумма = ' + sum);
         end;
      Console.WriteLine('Минимальная сумма = ' + min);
      Console.WriteLine('Максимальная сумма = ' + max);
      Console.WriteLine();
end;
```



Минимальное значение суммы сумм равно 73, а максимальное – 87

Тогда сумма сумм всех чисел находится в диапазоне:

73 <= s1 * 8 + 28 <= 87

Откуда получаем диапазон значений для первой суммы:

45 <= s1 * 8 <= 59
5 <= s1 <= 7

Мы знаем, что:

- в последовательности 9 чисел 1..9
- сумма первых двух чисел равна 5, 6 или 7
- следующая сумма на 1 больше предыдущей
- первым числом последовательности может быть любое число из ряда 1..s1-1

В **главном блоке** мы последовательно вызываем процедуру *Solve* с суммой первых двух чисел:

```
uses
   System;
type
   int = integer;

// Савин 3-76

begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 3.76';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Другая последовательность');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   //Solve();
   Solve(5);
   Solve(6);
   Solve(7);

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

В процедуре **Solve** мы начинаем последовательность со всех возможных чисел:

```
// РЕШАЕМ ЗАДАЧУ
procedure Solve(summa: int);
begin
```

```
   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Yellow;

   // первое число последовательности:
   for var i := 1 to summa - 1 do
   begin
      var sum := summa;
```

Помещаем первое число в список numbers, который нужен нам для проверки уже вышедших чисел:

```
 // список чисел:
 var numbers := new List<int>(Seq(i));
 var next := i;
```

Следующее число последовательности легко найти, вычтя текущее число из текущей же суммы:

```
while (true) do
begin
   // следующее число:
   next := sum - next;
```

Оно должно находиться в диапазоне 1..9 и отличаться от уже найденных чисел:

```
if ((next < 1) or (next > 9)) then
   break;
if (numbers.Contains(next)) then
   break;
numbers.Add(next);
```

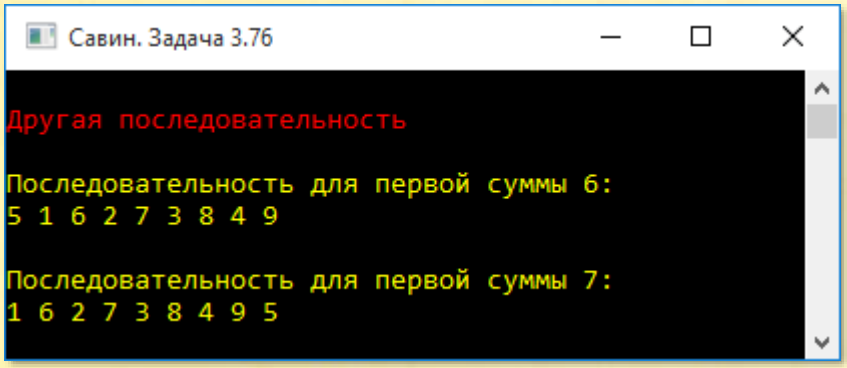
Для следующей пары чисел сумма увеличивается на 1:

```
  sum += 1;
end;
```

Рано или поздно бесконечный цикл *while* прервётся оператором *break*. Если в списке numbers окажется ровно 9 разных чисел из диапазона *1..9*, то решение задачи найдено, и мы печатаем его на экране:

```
      if (numbers.Count = 9) then
      begin
         Console.WriteLine('Последовательность для первой
                            суммы {0}: ', summa);
         foreach var num in numbers do
            Console.Write(num + ' ');
         Console.WriteLine();
      end
   end;
   Console.WriteLine();
end;
```

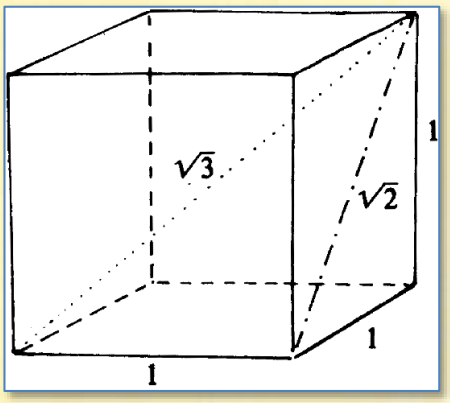
Как показывает рисунок, для первой суммы 5 последовательности не существует, а для сумм 6 и 7 наша программа нашла по одной последовательности:



Первую последовательность можно даже свернуть в кольцо!

А что же в книге? – В ответе присутствует только первая последовательность, которая почему-то начинается с нуля, хотя в условии задачи о нуле нет ни единого слова.

Кубический корень



> *У какого числа количество его тысяч равняется кубическому корню из этого числа?*

Минимальный кубический корень равен:

$$\sqrt[3]{1000}$$

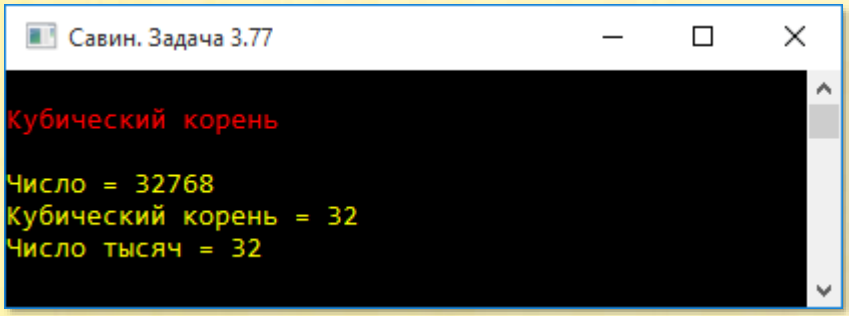
Так как «количество тысяч» - **целое** число, то искомое число можно найти, возводя целые числа, б*о*льшие минимального (или равное ему), в куб. Затем мы должны сравнить «количество тысяч» с корнем кубическим из искомого числа.

Задача легко решается простым перебором в бесконечном цикле *for*:

```
uses
   System;

type
   int = integer;

// Савин 3-77

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
```

```
   Console.ForegroundColor := ConsoleColor.Yellow;

   var min := Trunc(Math.Pow(1000, 1.0 / 3));
   for var n := min to int.MaxValue do
   begin
      // куб:
      var cube := n * n * n;
      // число тысяч:
      var n1000 := cube div 1000;
      if (n1000 = n) then
      begin
         Console.WriteLine('Число = ' + cube);
         Console.WriteLine('Кубический корень = ' + n);
         Console.WriteLine('Число тысяч = ' + n1000);
         Console.WriteLine('');
         break;
      end
   end;
   Console.WriteLine();
end;


begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 3.77';
   Console.WriteLine('');
   Console.ForegroundColor := ConsoleColor.Red;
   Console.WriteLine('Кубический корень');
   Console.ForegroundColor := ConsoleColor.Green;
   Console.WriteLine();

   Solve();

   Console.WriteLine();
   Console.ForegroundColor := ConsoleColor.Red;
end.
```

Искомое число вы видите на рисунке:



Кто первый?

В четвёртой главе книги собраны задачи в картинках:

Нулевая задача – логическая по форме и комбинаторная по содержанию. Но мы, чтобы не усугублять решение задачи комбинаторными объектами, прибегнем к простому **перебору** вариантов.

Нам известно, что Серёжа, Коля, Ваня, Надя и Толя заняли разные призовые места – от 1 до 5.

Так как каждый из них мог занять места с 1 по 5, то мы организуем 5 вложенных циклов, не забывая при этом выполнять условие, что все участники заняли **разные** места.

В самом глубоко вложенном цикле мы проверяем утверждения болельщиков.

Так как одно и только одно из них верно, то проверки удобно проводить с помощью логической операции **XOR**.

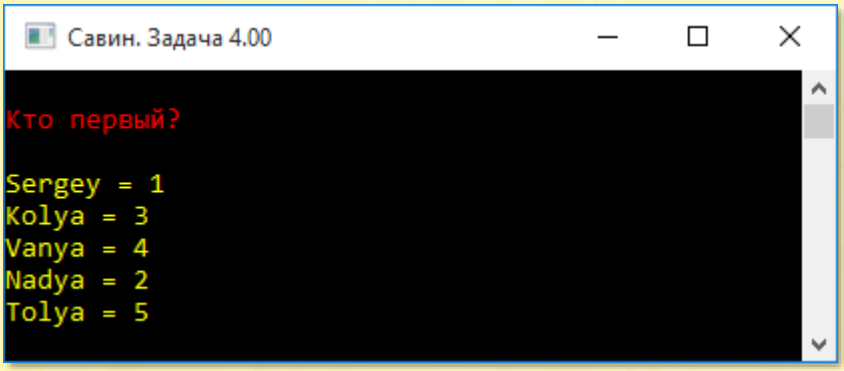
Осталось воплотить теорию в практику:

```
uses
   System, System.Linq;

type
   int = integer;

// Савин 4-00

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   // список мест:
   var numbers := Enumerable.Range(1, 5);
   foreach var Sergey in numbers do
   begin
      var numbers2 := numbers.ToList();
      numbers2.Remove(Sergey);
      foreach var Kolya in numbers2 do
      begin
         var numbers3 := numbers.ToList();
```

```
            numbers3.Remove(Sergey);
            numbers3.Remove(Kolya);
            foreach var Vanya in numbers3 do
            begin
               var numbers4 := numbers.ToList();
               numbers4.Remove(Sergey);
               numbers4.Remove(Kolya);
               numbers4.Remove(Vanya);
               foreach var Nadya in numbers4 do
               begin
                  var numbers5 := numbers.ToList();
                  numbers5.Remove(Sergey);
                  numbers5.Remove(Kolya);
                  numbers5.Remove(Vanya);
                  numbers5.Remove(Nadya);
                  foreach var Tolya in numbers4 do
                  begin
                     // проверяем высказывания:
                     if (not ((Sergey = 2) xor (Kolya = 3))) then
                        continue;
                     if (not ((Sergey = 2) xor (Vanya = 4))) then
                        continue;
                     if (not ((Nadya = 3) xor (Tolya = 5))) then
                        continue;
                     if (not ((Tolya = 1) xor (Nadya = 2))) then
                        continue;
                     if (not ((Kolya = 1) xor (Vanya = 4))) then
                        continue;
                     Console.WriteLine('Sergey = ' + Sergey);
                     Console.WriteLine('Kolya = ' + Kolya);
                     Console.WriteLine('Vanya = ' + Vanya);
                     Console.WriteLine('Nadya = ' + Nadya);
                     Console.WriteLine('Tolya = ' + Tolya);
                     Console.WriteLine();
                  end
               end
            end
         end
      end
   end;


begin
   // заголовок окна:
   Console.Title := 'Савин. Задача 4.00';
```

```
    Console.WriteLine('');
    Console.ForegroundColor := ConsoleColor.Red;
    Console.WriteLine('Кто первый?');
    Console.ForegroundColor := ConsoleColor.Green;
    Console.WriteLine();

    Solve();

    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```

И получить ответ на задачу:



Три мушкетёра

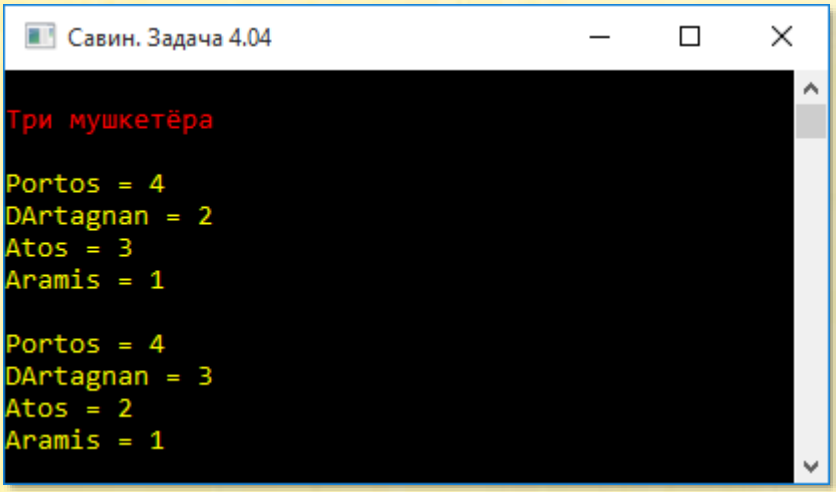
Эта задача ничем принципиально не отличается от предыдущей.

Обозначим силу мушкетёров в условных единицах числами от 1 до 4. Чем больше число, тем здоровее мушкетёр.

В задаче всего 3 логических условия, которые мы легко переведём на *паскаль*:

```
uses
   System, System.Linq;

type
   int = integer;

// Савин 4-04

// РЕШАЕМ ЗАДАЧУ
procedure Solve();
begin
   Console.ForegroundColor := ConsoleColor.Yellow;

   // список мест:
   var numbers := Enumerable.Range(1, 4);
   foreach var Portos in numbers do
   begin
      var numbers2 := numbers.ToList();
      numbers2.Remove(Portos);
      foreach var DArtagnan in numbers2 do
      begin
         var numbers3 := numbers.ToList();
         numbers3.Remove(Portos);
         numbers3.Remove(DArtagnan);
         foreach var Atos in numbers3 do
         begin
            var numbers4 := numbers.ToList();
            numbers4.Remove(Portos);
            numbers4.Remove(DArtagnan);
            numbers4.Remove(Atos);
            foreach var Aramis in numbers4 do
            begin
               // проверяем условия:
               if (not (Portos + DArtagnan > Atos + Aramis)) then
                  continue;
               if (not (Atos + Portos > Aramis + DArtagnan)) then
                  continue;
               if (not (Portos + Aramis = Atos + DArtagnan)) then
```

```
                    continue;

                Console.WriteLine('Portos = ' + Portos);
                Console.WriteLine('DArtagnan = ' + DArtagnan);
                Console.WriteLine('Atos = ' + Atos);
                Console.WriteLine('Aramis = ' + Aramis);
                Console.WriteLine();
            end
        end
    end
end;


begin
    // заголовок окна:
    Console.Title := 'Савин. Задача 4.04';
    Console.WriteLine('');
    Console.ForegroundColor := ConsoleColor.Red;
    Console.WriteLine('Три мушкетёра');
    Console.ForegroundColor := ConsoleColor.Green;
    Console.WriteLine();

    Solve();

    Console.WriteLine();
    Console.ForegroundColor := ConsoleColor.Red;
end.
```



А вот и 2 решения задачи:

В книге приводится только второй ответ. Очевидно, отгадку следует искать в первом условии, что Портос с д'Артаньяном легко перетянули Атоса с Арамисом. Для первого ответа сила Портоса с д'Артаньяном равна 6, для второго – 7.

Танцующие пары

Из условий задачи

Нам всем вместе 115 лет.
Каждая девушка моложе своего партнёра на 3 года.

следует, что общий возраст девушек равен:

(115 – 3 x 3)/2 = 53 года,

а их партнёрам 115 – 53 = 62 года.

Из условий, что все пары – из танцевального коллектива МФТИ и самая молодая девушка – Ира, следует, что ей не менее 16 лет.

Так как всес девушкам вместе 53 года, то Ире не больше 17 лет, иначе не выполнится условие, что она самая младшая.

Из условий

Меня зовут Юля. Мне вместе с Игорем 36 лет.
Я – Антон. Мне вместе с Юлей 40 лет.

следует, что Антон старше Игоря на 4 года.

Одна из девушек младше Антона на 3 года, а вторая младше Игоря на 3 года. Значит, первая старше второй на 4 года.

Пусть Ире 17 лет, тогда Светлане и Юле вместе – 36 лет. Одна из них по крайней мере на 4 года старше Иры, поэтому ей не менее 21 года. И тогда на долю второй девушки остаётся 15 лет, что невозможно. Итак, Ире 16 лет.

Светлане и Юле вместе – 37 лет. Ни одна из них не младше 17 лет. Если одной из них 18 лет, то второй 19 – и ни одна из девушек не старше другой на 4 года. Вывод: одной из девушек 17 лет, а второй – 20.

Тогда Игорю или 19 лет, или 16. Второй возраст не удовлетворяет условию задачи, поэтому Игорю 19 лет, а Юле 36 – 19 = 17 лет.

Мы установили, что Игорю 19 лет, а Юле – 17 лет.

Таким образом, Светлане 20 лет.

Партнёру Иры 19 лет. Это Игорь.

Антон на 4 года старше Игоря – ему 23 года. Он партнёр Светланы.

Партнёру Юли 21 год. Это Максим.

Задача решена логическим путём, без перебора, поэтому компьютер нам не понадобился.

Семёрку – в конец

Исходный код программы находится в папке **Савин →1.30**.

Трёхзначное число начинается с цифры 7. Из него получили другое трёхзначное число, переставив эту цифру в конец числа. Полученное число оказалось на 117 меньше предыдущего.

Какое число получилось?

Мы решим задачу двумя способами, но используем при этом один и тот же алгоритм.

«Классическое» решение может быть таким:

```
// РЕШАЕМ ЗАДАЧУ 1-30: Семёрку - в конец
procedure frmMain.btn130_Click(sender: Object; e: EventArgs);
begin
   var puzzle := new Savin_130(rtxRes);
   puzzle.Solve();
   puzzle.Solve2();
end;


unit Savin_130Unit;

uses System, System.Windows.Forms, System.Linq;
```

```
type Savin_130 = class
    rtxRes: RichTextBox;

    public constructor Create(rtx: RichTextBox);
    begin
        rtxRes := rtx;
    end;

    procedure Solve();
    begin
        var nVar := 0;
        for var i := 10 to 99 do
        begin
            var num1 := 700 + i;
            var num2 := i * 10 + 7;

            if (num2 + 117 = num1) then
            begin
                rtxRes.AppendText('Первое число = ' + num1);
                rtxRes.AppendText(Environment.NewLine);
                rtxRes.AppendText('Второе число = ' + num2);
                rtxRes.AppendText(Environment.NewLine);
                nVar += 1;
            end
        end;
        rtxRes.AppendText('Всего вариантов: ' + nVar);
        rtxRes.AppendText(NewLine + NewLine);
    end;
```

Из ответа в книге мы знаем, что решение единственное, но почему бы и не проверить это утверждение?

Задача очень простая, если быстро догадаться, что оба числа нужно записать в таком виде:

```
var num1 := 700 + i;
var num2 := i * 10 + 7;
```

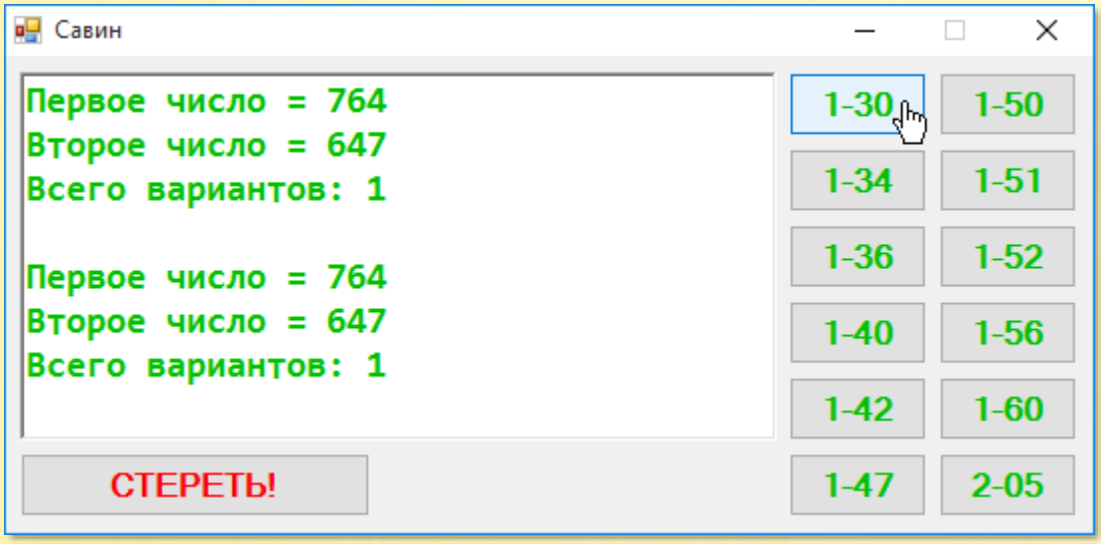
Второй способ связан с *LINQ*. Вместо цикла мы создаём последовательность чисел *10..99*:

```
procedure Solve2();
begin
    var nums := Enumerable.Range(10, 90);
```

Теперь выбираем из неё те числа, которые удовлетворяют условию задачи, и сразу формируем строку результата:

```
        var res := nums.Where(n -> n * 10 + 7 + 117 = 700 + n)
                .Select(n -> String
                .Format('Первое число = {0}' + NewLine +
                        'Второе число = {1}' + NewLine,
                       700 + n, n * 10 + 7));

         foreach var s in res do
         begin
            rtxRes.AppendText(s);
         end;
         rtxRes.AppendText('Всего вариантов: ' + res.Count());
         rtxRes.AppendText(NewLine);
      end;
   end;
end.
```

Рисунок показывает, что, каким способом ни решай задачу, а ответ у неё – *единственный*!



Возраст братьев

Исходный код программы находится в папке **Савин →1.34**.

В 1987 году возраст старшего из моих братьев стал равен сумме цифр года рождения младшего, а возраст младшего – сумме цифр года рождения старшего брата.

Сколько им было лет, если один старше другого на 7 лет?

Пусть младший брат родился в **19ab** году, а старший – в **19cd** году.

Тогда в 1987 году младшему брату исполнилось **1987 – 19ab = 87 – (a * 10 + b)** лет.

А старшему брату в 1987 году исполнилось **1987 – 19cd = 87 – (c * 10 + d)** лет.

С другой стороны, возраст младшего брата равен **1 + 9 + c + d**, а старшего – **1 + 9 + a + b**.

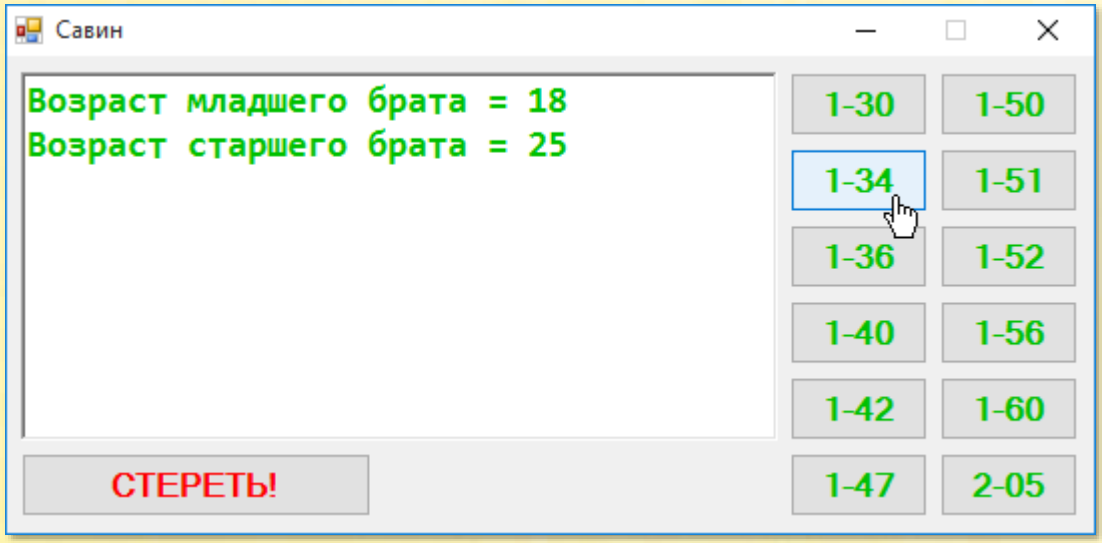
Этих рассуждений вполне достаточно, чтобы решить задачу простым **перебором** четырёх цифр:

```
unit Savin_134Unit;

uses
   System, System.Windows.Forms, System.Linq;

type
   Savin_134 = class
      rtxRes: RichTextBox;

      public constructor Create(rtx: RichTextBox);
      begin
         rtxRes := rtx;
      end;
```

```
        procedure Solve();
        begin
           var digits := Enumerable.Range(0, 10);
           foreach var a in digits do
              foreach var b in digits do
                 foreach var c in digits do
                    foreach var d in digits do
                    begin
                       var old := 1 + 9 + a + b;
                       var young := 1 + 9 + c + d;
                       if ((old = young + 7) and
                           (old = 87 - (c * 10 + d)) and
                           (young = 87 - (a * 10 + b))) then
                       begin
                          rtxRes.AppendText('Возраст младшего брата = '
                                            + young);
                          rtxRes.AppendText(Environment.NewLine);
                          rtxRes.AppendText('Возраст старшего брата = '
                                            + old);
                          rtxRes.AppendText(Environment.NewLine);
                       end
                    end
        end;
    end;

end.
```

Эту задачу можно решить иначе – попробуйте!

А **ответ** вы можете видеть на рисунке:



Сколько мне лет?

Исходный код программы находится в папке **Савин → 1.36.**

Дядя Алёша вдвое старше меня, а цифры числа, выражающего мой возраст, равны сумме и разности цифр его возраста.

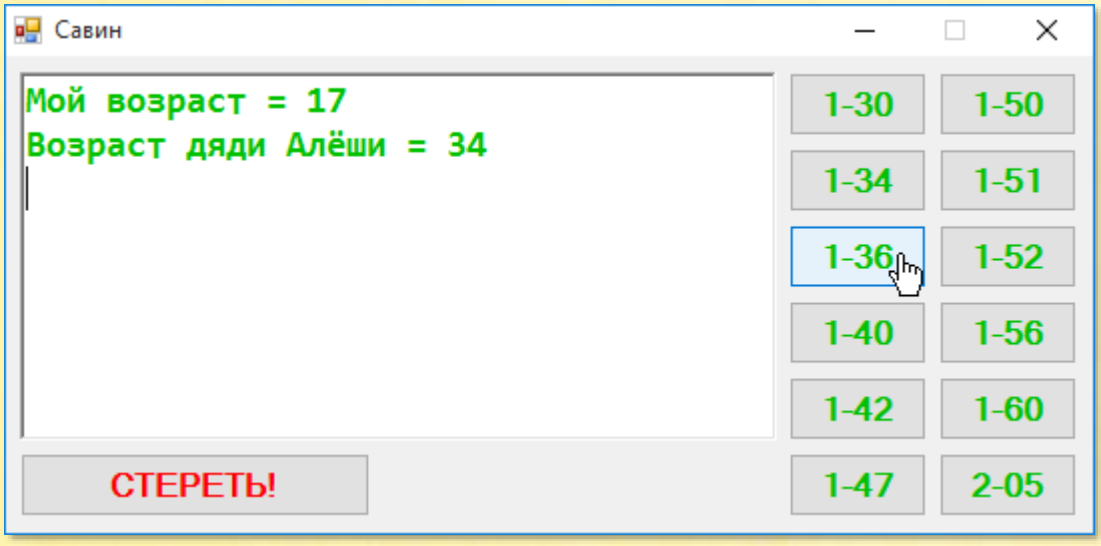
Сколько мне лет?

Из условия задачи следует, что мне (автору задачи) не меньше 10 лет. Чтобы решить задачу, достаточно **перебирать** цифры, составляющие мой возраст и сверяться с текстом задачи:

```
unit Savin_136Unit;

uses
   System, System.Windows.Forms, System.Linq;

type
   Savin_136 = class
      rtxRes: RichTextBox;

      public constructor Create(rtx: RichTextBox);
      begin
         rtxRes := rtx;
      end;

      procedure Solve();
      begin
         var digits := Enumerable.Range(0, 10);
```

```
            foreach var a in digits do
               foreach var b in digits do
               begin
                  var i := a * 10 + b;
                  // мой возраст - двузначное число:
                  if (i < 10) then
                     continue;
                  var onkel := i * 2;
                  var dig1 := onkel div 10;
                  var dig2 := onkel mod 10;
                  if (((dig1 + dig2 = a) and
                      (Math.Abs(dig1 - dig2) = b)) or
                      ((dig1 + dig2 = b) and
                      (Math.Abs(dig1 - dig2) = a))) then
                  begin
                     rtxRes.AppendText('Мой возраст = ' + i);
                     rtxRes.AppendText(Environment.NewLine);
                     rtxRes.AppendText('Возраст дяди Алёши = ' + onkel);
                     rtxRes.AppendText(Environment.NewLine);
                  end
               end
         end;

end.
```

К сожалению, задача не сильно развивает серое вещество, но для упражнения в программировании вполне годится:



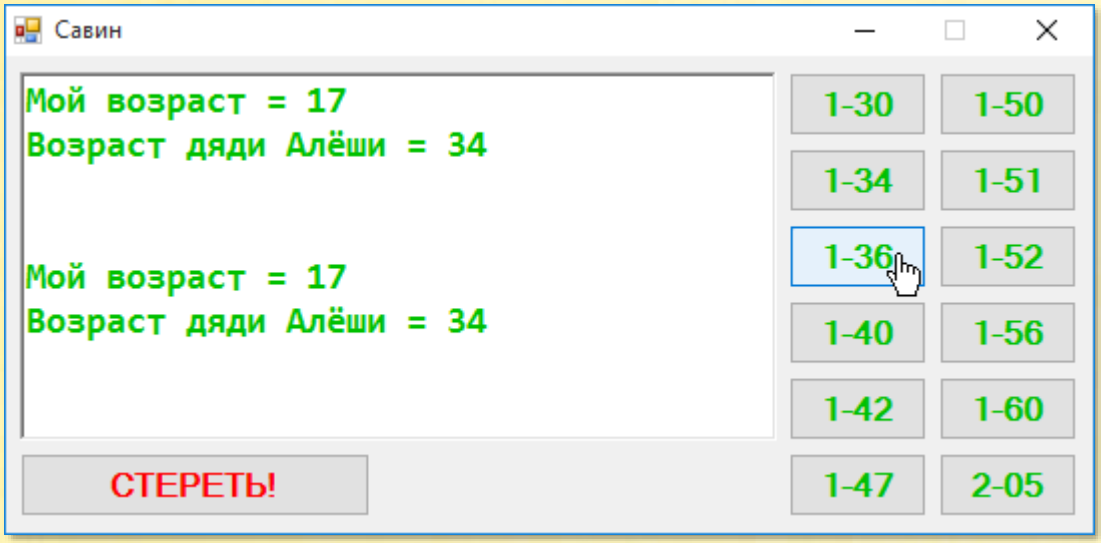
И уж коли речь зашла о тренировке в программировании, то давайте решим задачу с отягощениями.

Пусть мой возраст v изменяется от 10 лет до 49 - чтобы дядя Алёша имел также двузначный возраст.

Тогда возраст дяди равен v * 2. С помощью функций D1 и D2 мы легко найдём цифры его возраста, а с помощью функций A и B – цифры моего возраста. Теперь мы легко перепишем условие задачи в методе *Where*:

```
procedure Solve2();
begin
   var D1: IntFunc := v -> 2 * v div 10;
   var D2: IntFunc := v -> 2 * v mod 10;
   var A: IntFunc := v -> v div 10;
   var B: IntFunc := v -> v mod 10;
   var vozr := Enumerable.Range(10, 40);

   var res := vozr.Where(v -> (D1(v) + D2(v) = A(v)) and
                (Math.Abs(D1(v) - D2(v)) = B(v)) or
                (D1(v) + D2(v) = B(v)) and
                (Math.Abs(D1(v) - D2(v)) = A(v)));

   rtxRes.AppendText(NewLine + NewLine);
   foreach var r in res do
   begin
      rtxRes.AppendText('Мой возраст = ' + r);
      rtxRes.AppendText(Environment.NewLine);
      rtxRes.AppendText('Возраст дяди Алёши = ' + r * 2);
      rtxRes.AppendText(Environment.NewLine);
   end
end;
```

Ответ мы получаем, естественно, тот же самый, а решение задачи, несмотря на функции-делегаты и методы расширения, получилось даже более понятным, чем в первом случае!



Не бойтесь простых решений!

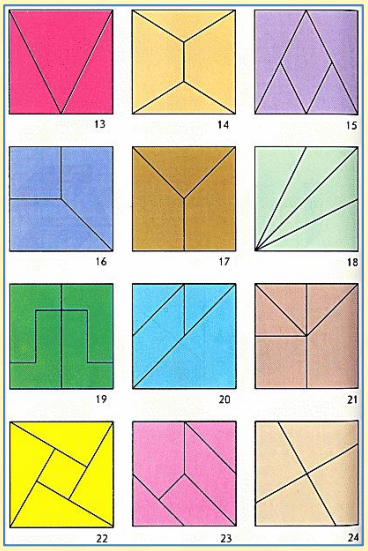
И так и сяк - квадрат

Исходный код программы находится в папке **Савин →1.40**.

Если к числу 20 прибавить 16, то получим 36 – полный квадрат. Если отнимем от него 16, то получим 4 – тоже полный квадрат.

Существуют ли ещё целые числа, которые становятся квадратами как после прибавления к ним 16, так и после вычитания из них 16?

Сколько их?



Последний вопрос задачи подсказывает нам, что таких чисел немного, иначе вопрос был бы неуместен. Это значит, что искать решения нужно только среди маленьких чисел.

Чтобы наша программа была более универсальной, сделаем прибавочное число параметром процедуры Solve, тогда, задавая различные значения этому параметру, мы сможем слегка генерализовать задачу.

Ограничим свои поиски числами *2..10000*:

```
procedure Solve(dob: int);
begin
   // числа:
   var nums := Enumerable.Range(2, 10000);
```

Мы сильно облегчим себе задачу по проверке чисел на «полную квадратность», если поместим нужное число квадратов в последовательность quads:

```
// квадраты чисел:
var quads := Range(2, 100).Select(n -> n * n);
```

Все числа, которые удовлетворяют условиям задачи, пополнят список res:

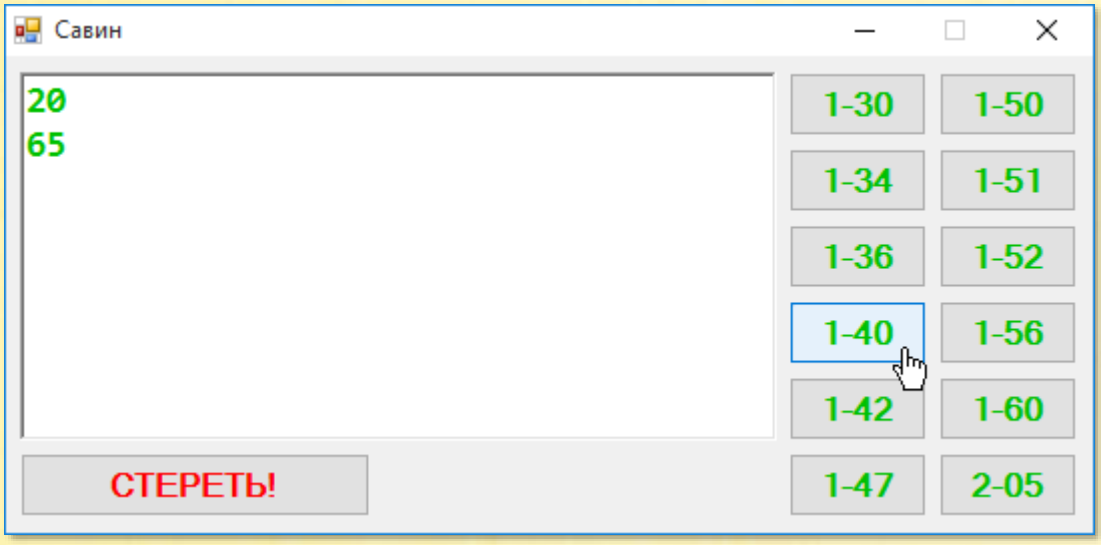
```
// Решаем задачу:
var res := nums.Where(n -> quads.Contains(n - dob) and
                           quads.Contains(n + dob));
```

После чего мы напечатаем их на экране:

```
   // печатаем ответ:
   foreach var r in res do
   begin
      rtxRes.AppendText(r.ToString());
      rtxRes.AppendText(Environment.NewLine);
   end;
```

```
   rtxRes.AppendText(NewLine + NewLine);
end;
```

Рисунок подтверждает книжный ответ: кроме числа 20, существует только одно число с такими же свойствами. Это 65.



Задавая различные значения параметру в методе *Solve*:

```
procedure frmMain.btn140_Click(sender: Object; e: EventArgs);
begin
   var puzzle := new Savin_140(rtxRes);
   puzzle.Solve(16);
end;
```

можно найти ещё несколько подобных чисел:

```
  45 - 36 = 9
  45 + 36 = 81

  85 - 36 = 49
  85 + 36 = 121

 325 - 36 = 289
```

```
 325 + 36 = 361

  80 - 64 = 16
  80 + 64 = 144

 260 - 64 = 196
 260 + 64 = 324

1025 - 64 = 961
1025 + 64 = 1089

125 - 100  = 25
125 + 100 = 225

629 - 100 = 529
629 + 100 = 729

2501 - 100 = 2401
2501 + 100 = 2601
```

Задача Савина

Если мы обозначим добавочное число как **dob**, исходное число через **n**, а искомые квадраты через **q1** и **q2**, то условие задачи можно записать так:

```
n – dob = q1
n + dob = q2
```

Откуда получаем:

```
2 * dob = q2 – q1
```

Так как в этом равенстве все числа – полные квадраты, то его можно переписать иначе:

$$2 * dob^2 = q2^2 - q1^2 = (q2 - q1) * (q2 + q1)$$

Если мы будем изменять переменные *dob* и *q1* в некотором диапазоне, то q2 в квадрате можно вычислить по формуле:

$$q2^2 = 2 * dob^2 + q1^2$$

А исходное число – по формуле:

$$n = q1^2 + dob^2$$

Чтобы не усложнять предыдущий проект, мы начнём новый и решим задачу Савина для различных значений добавочного числа. А его верхнюю границу мы получим от элемента управления udMax типа *NumericUpDown*.

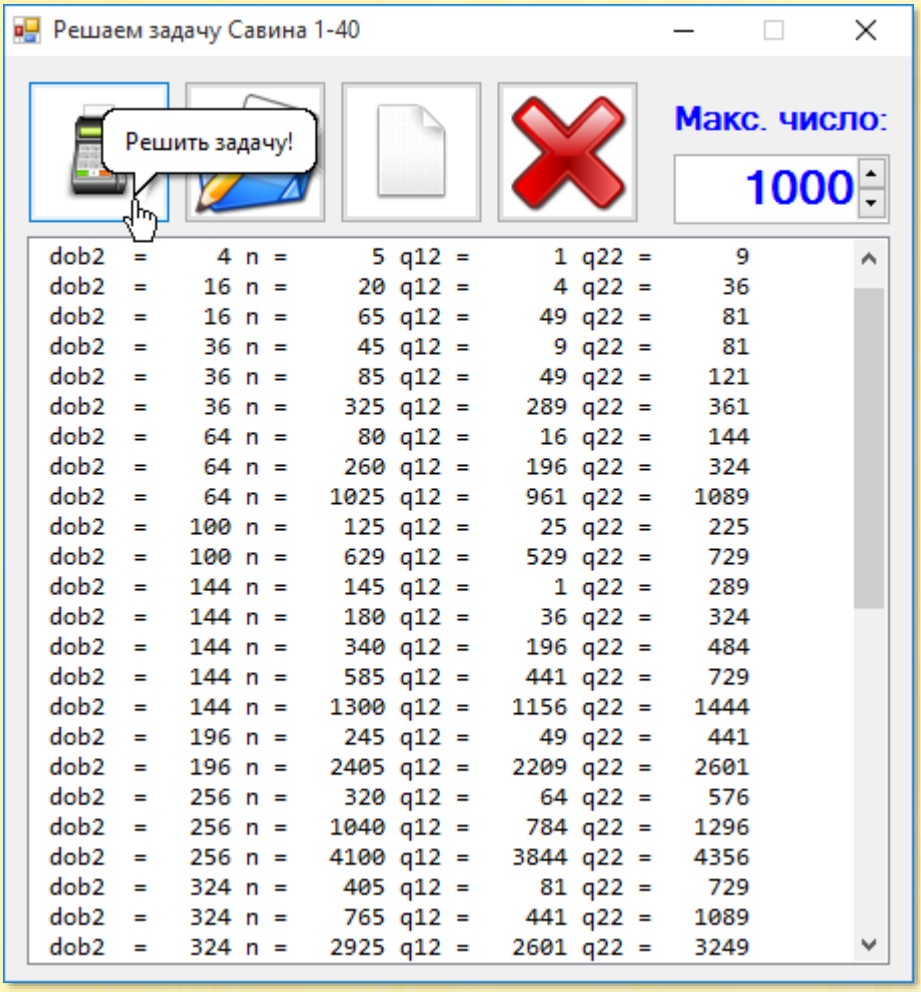
А теперь нажимаем кнопку btnSolve и отправляемся решать задачу:

```
implementation

var lst := new List<string>();

procedure Form1.btnSolve_Click(sender: Object; e: EventArgs);
begin
   // макс. число для проверки:
   var max := integer(udMax.Value);
   lst.Clear;
   Savin140(max);
   lstRes.TopIndex := lstRes.Items.Count - 23;
end;
```

Метод Savin140 строго следует нашим формулам, но число решений напрямую зависит от заданного диапазона для добавочного числа *dob*, а также от диапазона для чисел *q1*:

```
procedure Form1.Savin140(max: integer);
begin
   // квадраты чисел:
   var quads := Enumerable.Range(1, 4 * max).Select(n -> n * n);
   var nVar := 0;

   // перебираем значения dob в заданном диапазоне:
   var dob := 1;
   while(dob * dob <= max)  do
   begin
      var dob2 := dob * dob;
      var q1 := 1;
      while(q1 * q1 <= 16 * dob2) do
      begin
         var q12 := q1 * q1;
         if (quads.Contains(2 * dob2 + q12)) then
         begin
            var s := string.Empty;
            s += ' dob2  = ' + dob2.ToString().PadLeft(5);
            s += ' n = ' + (dob2 + q12).ToString().PadLeft(6);
            s += ' q12 = ' + (q12).ToString().PadLeft(6);
            s += ' q22 = ' + (2 * dob2 + q12).ToString().PadLeft(6);

            lstRes.Items.Add(s);
            lst.Add(s);
            nVar += 1;
         end;
         q1 += 1;
      end;
      dob += 1;
   end;
   lstRes.Sorted := true;
   lstRes.Items.Add('Всего ' + nVar);
   lstRes.Items.Add('');
end;
```

Совершенно очевидно, что все решения общей задачи найти невозможно, поэтому мы ограничимся только небольшими числами, которые ещё можно проверить в уме:


Решаем задачу Савина 1-40
Решить задачу!
Макс. число:
1000
dob2 = 4 n = 5 q12 = 1 q22 = 9
dob2 = 16 n = 20 q12 = 4 q22 = 36
dob2 = 16 n = 65 q12 = 49 q22 = 81
dob2 = 36 n = 45 q12 = 9 q22 = 81
dob2 = 36 n = 85 q12 = 49 q22 = 121
dob2 = 36 n = 325 q12 = 289 q22 = 361
dob2 = 64 n = 80 q12 = 16 q22 = 144
dob2 = 64 n = 260 q12 = 196 q22 = 324
dob2 = 64 n = 1025 q12 = 961 q22 = 1089
dob2 = 100 n = 125 q12 = 25 q22 = 225
dob2 = 100 n = 629 q12 = 529 q22 = 729
dob2 = 144 n = 145 q12 = 1 q22 = 289
dob2 = 144 n = 180 q12 = 36 q22 = 324
dob2 = 144 n = 340 q12 = 196 q22 = 484
dob2 = 144 n = 585 q12 = 441 q22 = 729
dob2 = 144 n = 1300 q12 = 1156 q22 = 1444
dob2 = 196 n = 245 q12 = 49 q22 = 441
dob2 = 196 n = 2405 q12 = 2209 q22 = 2601
dob2 = 256 n = 320 q12 = 64 q22 = 576
dob2 = 256 n = 1040 q12 = 784 q22 = 1296
dob2 = 256 n = 4100 q12 = 3844 q22 = 4356
dob2 = 324 n = 405 q12 = 81 q22 = 729
dob2 = 324 n = 765 q12 = 441 q22 = 1089
dob2 = 324 n = 2925 q12 = 2601 q22 = 3249

Математик и ГАИ

Исходный код программы находится в папке **Савин →1.42**.

К задумчиво стоящему на тротуаре математику подъехал милиционер. «Вы не обратили внимания на номер проехавшего сейчас самосвала?» - спросил он.

«О да! У него редкостный номер! Второе двузначное число получается из первого перестановкой цифр, а их разность равна сумме цифр каждого из них».

Какой же номер у самосвала?

Поскольку первое число - двузначное, то мы можем искать его среди всех двузначных чисел, тогда второе число легко вычислить в функции N2, а сумму цифр – в функции S:

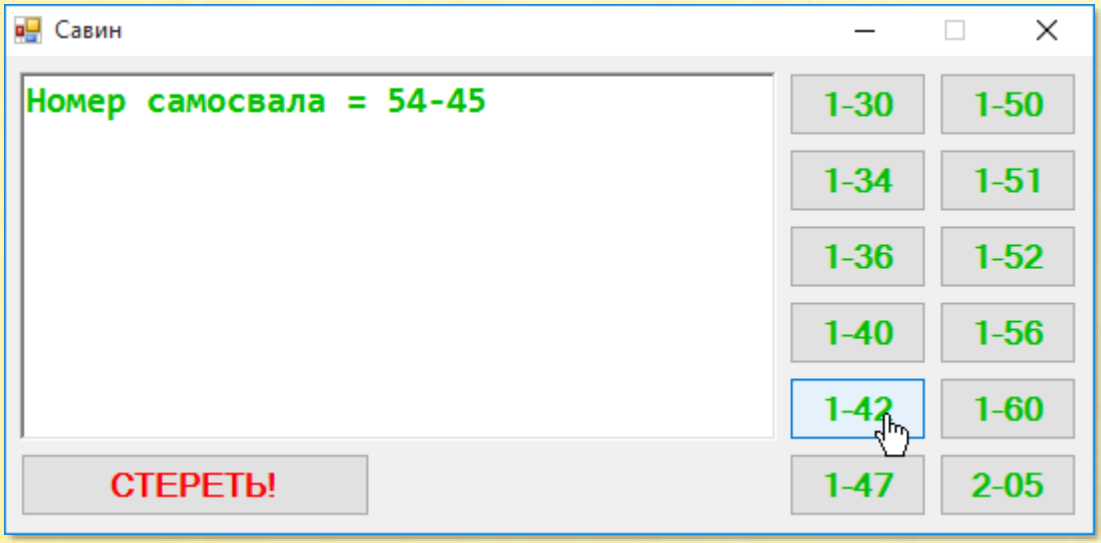
```
unit Savin_142Unit;

uses
   System, System.Windows.Forms, System.Linq;

type
   int = integer;

type
   Savin_142 = class
      rtxRes: RichTextBox;

      public constructor Create(rtx: RichTextBox);
      begin
         rtxRes := rtx;
      end;

      procedure Solve();
```

```
    begin
      // второе число:
      var N2: Func<int, int> := n -> n mod 10 * 10 + n div 10;
      // сумма цифр:
      var S: Func<int, int> := n -> n div 10 + n mod 10;

      // первое число:
      var n1 := Enumerable.Range(12, 87);
```

И нам нужно найти числа, которые удовлетворяют условиям задачи:

```
  var res := n1.Where(n -> n - N2(n) = S(n));
  foreach var r in res do
  begin
    rtxRes.AppendText('Номер самосвала = ' + r + '-' + N2(r));
    rtxRes.AppendText(Environment.NewLine);
  end;

  rtxRes.AppendText(NewLine + NewLine);
end;
```

Как вы видите, задача решается небольшим перебором, а ответ на неё показан на рисунке:



Что за числа?

Исходный код программы находится в папке **Савин → 1.47**.

Найдите все такие двузначные числа, которые делятся на каждую из цифр в их записи.

С помощью функций **D1** и **D2** мы находим цифры двузначного числа, а функция **D** сообщает нам, делится ли заданное двузначное число на свои цифры:

```
procedure Solve();
begin
   // первая цифра:
   var D1: Func<int, int> := n -> n div 10;
   // вторая цифра:
   var D2: Func<int, int> := n -> n mod 10;
   // делим число на его цифры:
   var D: Func<int, bool> := n -> (n mod D1(n) = 0) and
                                  (n mod D2(n) = 0);
```

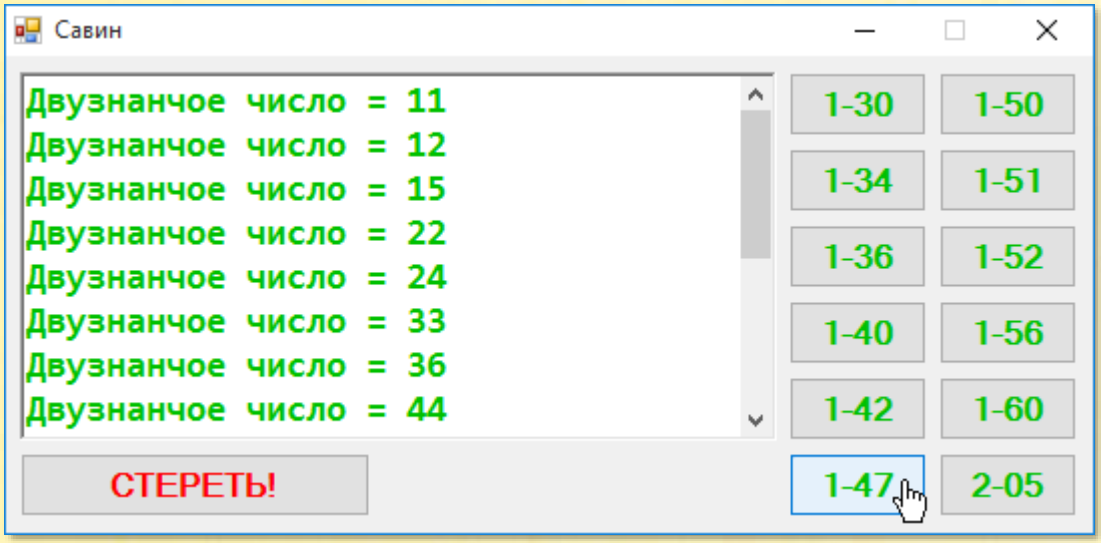
Так как строгие правила математики запрещают делить на нуль, то мы сразу игнорируем двузначные числа, заканчивающиеся нулём:

```
// двузначные числа без нуля:
var n2 := Range(11, 99 - 11 + 1).Where(n -> D2(n) <> 0);
```

Скрупулёзно и тщательно отбираем из списка те числа, которые разом делятся на обе свои цифры – и задача с двузначными числами однозначно решена!

```
   var res := n2.Where(n -> D(n));
```

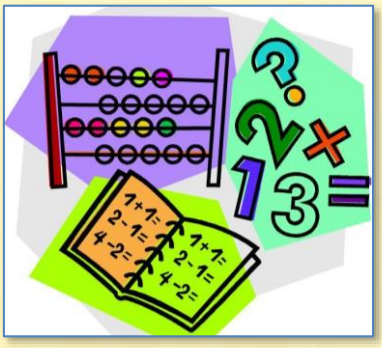
```
    foreach var r in res do
    begin
       rtxRes.AppendText('Двузначное число = ' + r);
       rtxRes.AppendText(Environment.NewLine);
    end;

    rtxRes.AppendText(NewLine + NewLine);
end;
```



Ищем число

Исходный код программы находится в папке **Савин** →1.50.

Существует ли трёхзначное число, делящееся на 11, у которого первая цифра больше второй, а вторая больше третьей?

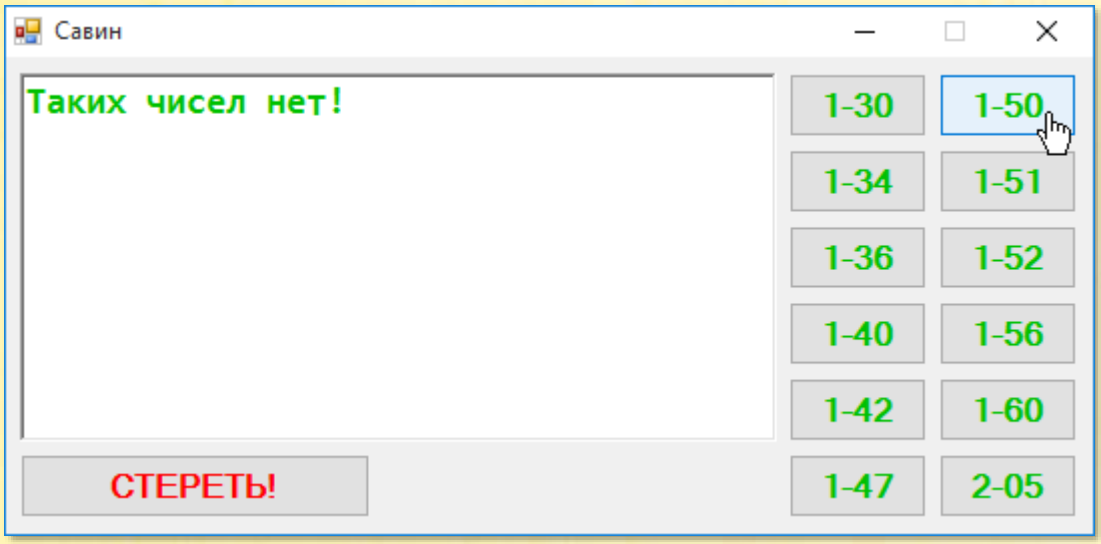


Задача легко решается **полным перебором** трёхзначных чисел и проверкой условий задачи:

```
unit Savin_150Unit;

uses
   System, System.Windows.Forms, System.Linq;

type
   int = integer;
   bool = boolean;

type
   Savin_150 = class
      rtxRes: RichTextBox;

      public constructor Create(rtx: RichTextBox);
      begin
         rtxRes := rtx;
      end;

      procedure Solve();
      begin
         // первая цифра:
         var D1: Func<int, int> := n -> n div 100;
         // вторая цифра:
         var D2: Func<int, int> := n -> n mod 100 div 10;
         // третья цифра:
         var D3: Func<int, int> := n -> n mod 10;

         // Трёхзначные числа, кратные 11:
         var n2 := Enumerable.Range(110, 1000 - 110)
                             .Where(n -> n mod 11 = 0);

         var res := n2.Where(n -> (D1(n) > D2(n)) and
                                  (D2(n) > D3(n)));
         if (res.Count() = 0) then
         begin
            rtxRes.AppendText('Таких чисел нет!');
            rtxRes.AppendText(Environment.NewLine);
         end;
         foreach var r in res do
         begin
            rtxRes.AppendText('Трёхзначное число = ' + r);
            rtxRes.AppendText(Environment.NewLine);
         end;
```

```
            rtxRes.AppendText(NewLine + NewLine);
        end;
    end;
end.
```

Как показывает рисунок, автор слегка поглумился над доверчивыми любителями математики – **решений у задачи нет** (если не считать решением его отсутствие).



Два простых

Исходный код программы находится в папке **Савин →1.51**.

Два двузначных простых числа получаются друг из друга перестановкой цифр, а их разность – полный квадрат.

Какие это числа?

В первую очередь, нам понадобится список простых чисел, среди которых и следует искать ответ на задачу. Метод GetPrimes поможет нам в этом:

```
unit Savin_151Unit;

uses
   System, System.Windows.Forms, System.Linq, System.Collections;

type
   int = integer;
   bool = boolean;

type
   Savin_151 = class
      rtxRes: RichTextBox;

      public constructor Create(rtx: RichTextBox);
      begin
         rtxRes := rtx;
      end;

      function GetPrimes(max: int): List<int>;
      begin
         var numbers := Range(3, max - 3);
         var primes := numbers
                  .Where(n -> Range(2, Trunc(Math.Sqrt(n)))
                  .All(i -> n mod i <> 0)).ToList();
         Result := primes;
      end;
```

Обратное число мы получим от функции Rev, а проверить число на его принадлежность к полным квадратам нам поможет функция IsQuadrat:

```
procedure Solve();
begin
   // обратное число:
   var Rev: Func<int, int> := n -> n mod 10 * 10 + n div 10;
   // проверяем заданное число на квадратность:
   var IsQuadrat: Func<int, bool> := n ->
   Trunc(Math.Sqrt(n) * Trunc(Math.Sqrt(n))) = n;
```

Отбираем из списка простых чисел только двузначные:

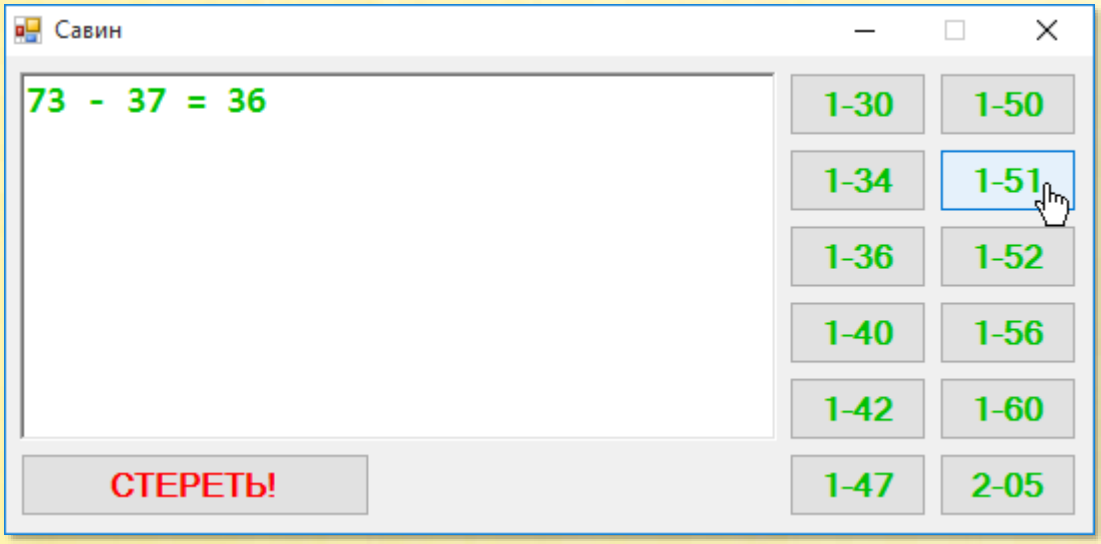
```
// простые двузначные числа:
var p2 := GetPrimes(99).Where(n -> n > 9);
```

И решаем задачу:

```
        var res := p2.Where(n -> p2.Contains(Rev(n)) and
                                 (n > Rev(n)) and
                                 IsQuadrat(n - Rev(n)));

         foreach var r in res do
         begin
            rtxRes.AppendText(r + ' - ' + Rev(r) +
                                ' = ' + Math.Abs(r - Rev(r)));
            rtxRes.AppendText(Environment.NewLine);
         end;
         rtxRes.AppendText(NewLine + NewLine);
      end;
   end;
end.
```

Здесь мы предполагаем, что первое число больше второго, иначе найдём каждую пару чисел дважды.

А рисунок подсказывает нам, что такая пара чисел – *единственная*:



Простой номер телефона

Исходный код программы находится в папке **Савин → 1.52**.

Алло, Катя! Нам поставили телефон! Запиши номер. Он, как и у тебя, пятизначный. Первая цифра – простое число, следующие 2 цифры - двузначное простое число, а последние 2 цифры получаются из предыдущей пары их перестановкой и образуют точный квадрат. Сумма же этих двух цифр равняется первой цифре номера.

Так какой же у меня номер телефона?

С чувством глубокой благодарности мы возьмём функции **Rev**, **S**, **IsQuadrat** и **GetPrimes** из наших предыдущих проектов и сразу приступим к решению задачи:

```
unit Savin_152Unit;

uses
   System, System.Windows.Forms, System.Linq;

type
   int = integer;
   bool = boolean;

type
   Savin_152 = class
      rtxRes: RichTextBox;

      public constructor Create(rtx: RichTextBox);
      begin
         rtxRes := rtx;
      end;
```

```
function GetPrimes(max: int): List<int>;
begin
   var numbers := Enumerable.Range(3, max - 3);
   var primes := numbers
           .Where(n -> Enumerable.Range(2, Trunc(Math.Sqrt(n)))
                       .All(i -> n mod i <> 0)).ToList();
   Result := primes;
end;

procedure Solve();
begin
   // обратное число:
   var Rev: Func<int, int> := n -> n mod 10 * 10 + n div 10;
   // проверяем заданное число на квадратность:
   var IsQuadrat: Func<int, bool> := n ->
   Trunc(Math.Sqrt(n) * Trunc(Math.Sqrt(n))) = n;
   // сумма цифр:
   var S: Func<int, int> := n -> n div 10 + n mod 10;
```

Первая цифра номера – однозначное простое число:

```
// простые однозначные числа:
var p1 := GetPrimes(9);
```

Следующие 2 цифры образуют двузначное простое число:

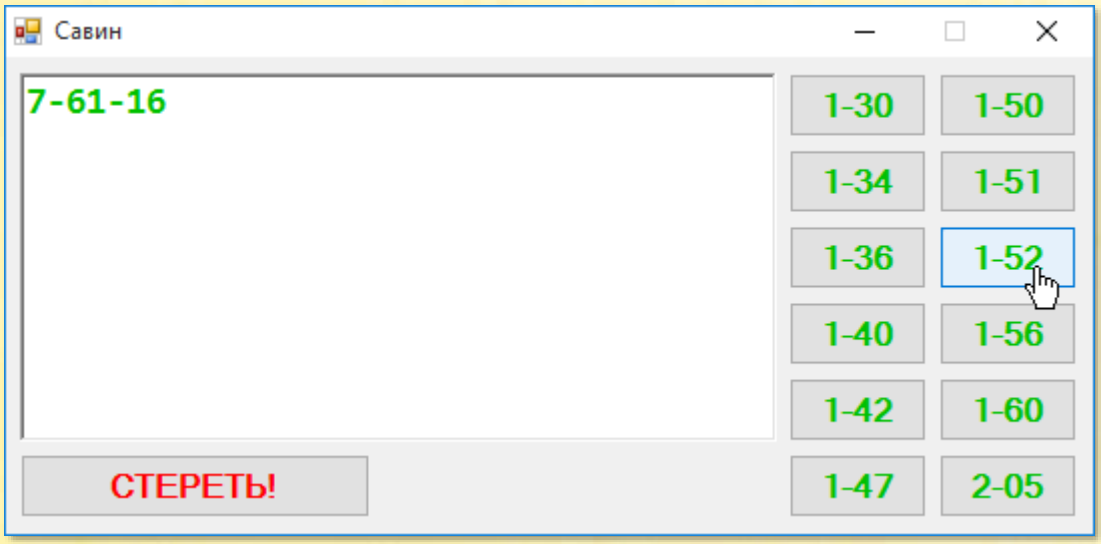
```
// простые двузначные числа:
var p2 := GetPrimes(99).Where(n -> n > 9);
```

Сумма цифр нужного нам простого двузначного числа должна быть однозначным простым числом, а его зеркальный двойник – полным квадратом:

```
// решаем задачу:
var res := p2.Where(p -> p1.Contains(S(p)) and
                         IsQuadrat(Rev(p)))
             .Select(p -> S(p) + '-' + p + '-' + Rev(p));
```

Как видите, при хорошей методической подготовке задача решается в два счёта!

```
            foreach var ss in res do
            begin
               rtxRes.AppendText(ss);
               rtxRes.AppendText(Environment.NewLine);
            end;

            rtxRes.AppendText(NewLine + NewLine);
         end;

   end;
end.
```



Девятку – в конец

Исходный код программы находится в папке **Савин → 1.56**.

Цифру 9, с которой начиналось трёхзначное число, перенесли в конец числа. В результате получилось число на 216 меньшее.

Какое число было первоначально?

Совершенно бесхитростная задача для искушённых решателей.

Но для большего облегчения и взаимопонимания мы напишем функцию **Trans**, чтобы комфортно переносить переднюю цифру из авангарда в арьергард:

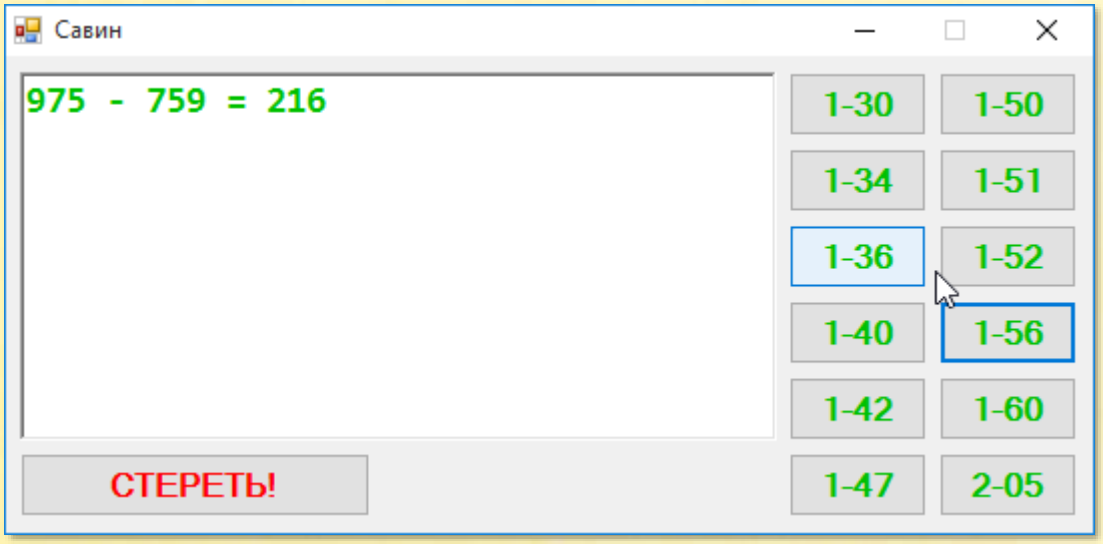
```
unit Savin_156Unit;

uses
   System, System.Windows.Forms, System.Linq;

type
   int = integer;
   bool = boolean;

type
   Savin_156 = class
      rtxRes: RichTextBox;

      public constructor Create(rtx: RichTextBox);
      begin
         rtxRes := rtx;
      end;


      procedure Solve();
      begin
```

```
        // первая цифра:
        var D1: Func<int, int> := n -> n div 100;
        // вторая цифра:
        var D2: Func<int, int> := n -> n mod 100 div 10;
        // третья цифра:
        var D3: Func<int, int> := n -> n mod 10;
        // переносим первую цифру в конец числа:
        var Trans: Func<int, int> := n -> D2(n) * 100 +
                                         D3(n) * 10 + D1(n);
        // трёхзначные числа:
        var n3 := Enumerable.Range(900, 999 - 900 + 1);
```

Выбираем из всех трёхзначных чисел, начинающихся с девятки те, которые на 216 больше «перенесённого» числа:

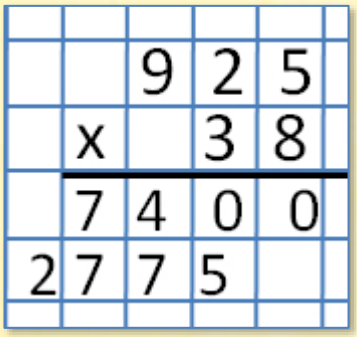
```
        // решаем задачу:
        var res := n3.Where(n -> n = Trans(n) + 216)
                     .Select(n -> n + ' - ' + Trans(n) + ' = ' +
                                  (n - Trans(n)));
        foreach var s in res do
        begin
           rtxRes.AppendText(s);
           rtxRes.AppendText(Environment.NewLine);
        end;
        rtxRes.AppendText(NewLine + NewLine);
    end;

  end;
end.
```

Правда, решение и на этот раз оказалось *единственным*:



Произведения цифр

Исходный код программы находится в папке **Савин →1.60**.



		9	2	5	
	x		3	8	
	7	4	0	0	
2	7	7	5		

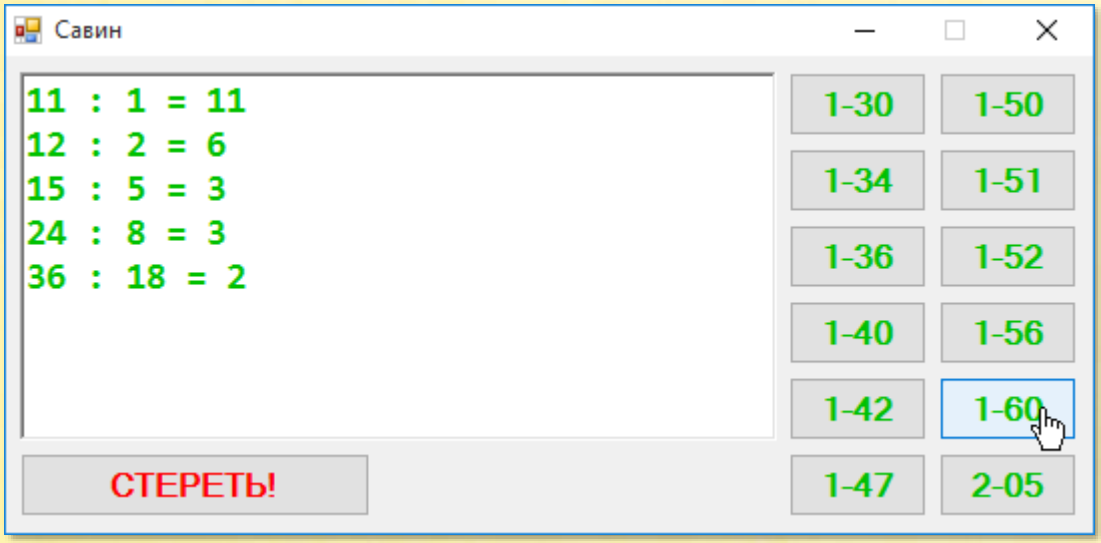
Найдите все двузначные числа, в целое число раз большие произведения своих цифр.

Самая большая изюминка и хитринка в этой задаче – избежать деления на нуль! Но если бдительность и прозорливость вас не подвели, то всё остальное – дело науки и техники:

```
unit Savin_160Unit;

uses
   System, System.Windows.Forms, System.Linq;

type
   int = integer;
   bool = boolean;

type
   Savin_160 = class
      rtxRes: RichTextBox;

      public constructor Create(rtx: RichTextBox);
      begin
         rtxRes := rtx;
      end;
```

```
      procedure Solve();
      begin
         // первая цифра:
         var D1: Func<int, int> := n -> n div 10;
         // вторая цифра:
         var D2: Func<int, int> := n -> n mod 10;
         // произведение цифр:
         var Mult: Func<int, int> := n -> D1(n) * D2(n);
         // двузначные числа:
         var n2 := Enumerable.Range(10, 100 - 10)
                             .Where(n -> D1(n) * D2(n) <> 0);
         // решаем задачу:
         var res := n2.Where(n -> n mod Mult(n) = 0)
                      .Select(n -> n + ' : ' + Mult(n) + ' = ' +
                             (n div Mult(n)));

         foreach var s in res do
         begin
            rtxRes.AppendText(s);
            rtxRes.AppendText(Environment.NewLine);
         end;
         rtxRes.AppendText(NewLine + NewLine);
      end;

   end;
end.
```

Весь скудный список двузначных чисел представлен на рисунке:



Как можно больше

Исходный код программы находится в папке **Савин →2.05**.

Впишите в квадратики цифры 1, 2, …, 9, каждую по одному разу так, чтобы произведение оказалось наибольшим.

Как видите, эта задача чисто **переборная**, поэтому решать её вручную не рекомендуется!

А нам – с помощью программы! – нужно **найти 3 трёхзначных числа из неповторяющихся цифр**.

Можно перебирать в длинном цикле все 9-значные числа от *123456789* до *987654321* и отбирать из них те, которые состоят из разных цифр:

```
for var n := 123456789 to 987654321 do
```

Затем разбить каждое 9-значное число на 3 трёхзначных и найти их произведение:

```
var N1: Func<int, int> := n -> n div 1000000;
var N2: Func<int, int> := n -> n mod 1000000 div 1000;
var N3: Func<int, int> := n -> n mod 1000;
var Mult: Func<int, int> := n -> N1(n) * N2(n) * N3(n);
```

Максимальное произведение и соответствующее ему 9-значное число нужно запомнить в переменных:

```
if (m > max) then
begin
   max := m;
   maxn := n;
end;
```

Но более продуктивный способ решения этой задачи состоит в том, чтобы генерировать **перестановки** из однозначных натуральных чисел, а уже из них формировать нужные трёхзначные числа.

Здесь нам, конечно, понадобится **класс**, который умеет генерировать перестановки. Мы не станем изобретать комбинаторный велосипед, а воспользуемся готовым решением с сайта *www.interact-sw.co.uk*.

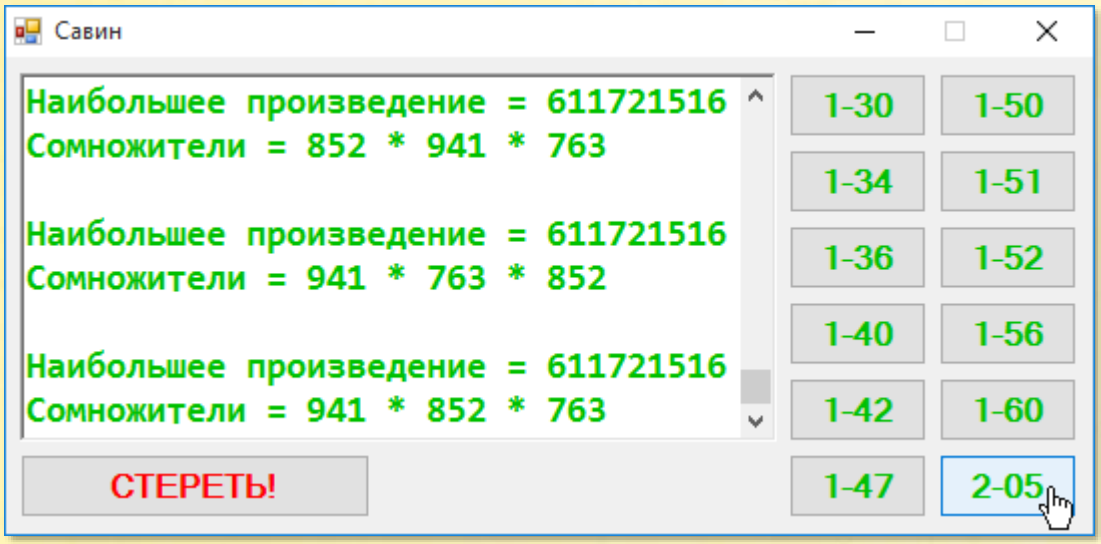
Класс **PermuteUtils** написан на языке *Си-шарп* и, к сожалению, не поддаётся переводу на *паскаль*, поэтому я создал динамическую библиотеку **PermuteUtils.dll**, которую необходимо подключить к проекту. Исходный код библиотеки вы можете найти в папке *PermuteUtils*.

Само решение задачи даётся нам без особого труда:

```
unit Savin_205Unit;

uses
   System, System.Windows.Forms, System.Linq, PermuteUtils;

type
   int = integer;
   bool = boolean;

type
   Savin_205 = class
      rtxRes: RichTextBox;

      public constructor Create(rtx: RichTextBox);
      begin
         rtxRes := rtx;
      end;
```

```
      // печатаем ответ
      procedure Print(max: int; mults: string);
      begin
         rtxRes.AppendText(String.Format('Наибольшее произведение =
                                          {0}', max));
         rtxRes.AppendText(Environment.NewLine);
         rtxRes.AppendText(String.Format('Сомножители = {0}', mults));
         rtxRes.AppendText(Environment.NewLine);
         rtxRes.AppendText(NewLine);
         rtxRes.ScrollToCaret();
      end;

      procedure Solve();
      begin
         // цифры:
         var digits := Enumerable.Range(1, 9);
         // макс. произведение:
         var max := 0;
         var mults := String.Empty;
         foreach var perm in PermuteUtils.PermuteUtils
                             .Permute(digits,9) do
         begin
            var n1 := perm.ElementAt(0) * 100 +
                      perm.ElementAt(1) * 10 +
                      perm.ElementAt(2);
            var n2 := perm.ElementAt(3) * 100 +
                      perm.ElementAt(4) * 10 +
                      perm.ElementAt(5);
            var n3 := perm.ElementAt(6) * 100 +
                      perm.ElementAt(7) * 10 +
                      perm.ElementAt(8);
            var m := n1 * n2 * n3;
            if (m >= max) then
            begin
               max := m;
               mults := n1 + ' * ' + n2 + ' * ' + n3;
               Print(max, mults);
            end;
         end;
         rtxRes.AppendText(NewLine + NewLine);
      end;

   end;
end.
```

На всякий пожарный случай мы по ходу решения задачи **печатаем** все текущие рекорды, а самый рекордный рекорд – *единственный*, если не считать перестановок сомножителей:



Задания для самостоятельного решения

Задача 4.01

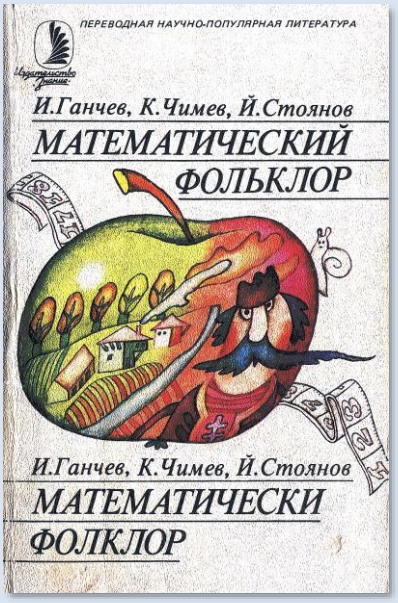
Сколько денег в кошельке

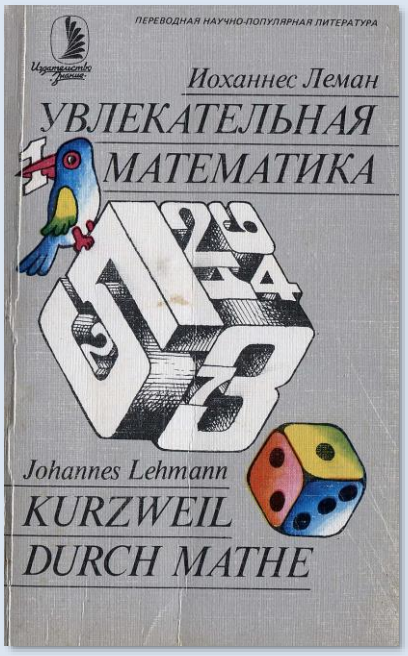
Решите задачу в картинках:

Задача легко решается ретроспективно, то есть задом наперёд.

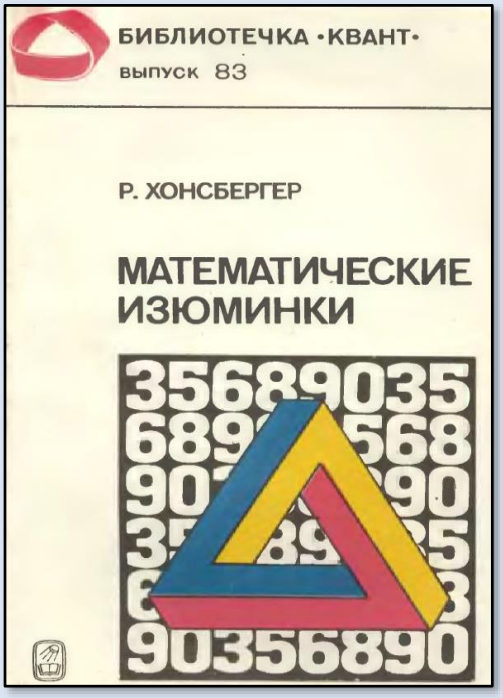
Ответ: 2 рубля 10 копеек.

Литература

[Фольклор]	
	Ганчев, Чимев, Стоянов Математический фольклор Знание. Москва, 1985. – 208 с.

[Увлекательная математика]	
	Иоханнес Леман Увлекательная математика Знание. Москва, 1985. – 272 с.

[XP92]



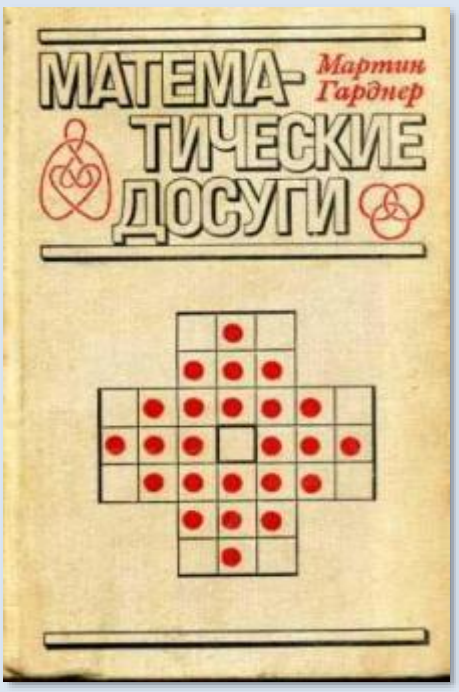
Росс Хонсбергер

Математические изюминки

Наука, 1992. - 176 с.
Библиотечка «Квант». Вып. 83

ISBN 5-02-014406-1

[ГМ72]



Гарднер Мартин

Математические досуги

М.:Мир, 1972. – 495 с.