МАТЕРИАЛЫ ЗАДАНИЙ ОТБОРОЧНОГО ЭТАПА

Олимпиады школьников "Надежда энергетики" в 2017/18 учебном году

ИНФОРМАТИКА

ЗАДАНИЕ ПО ИНФОРМАТИКЕ
ВАРИАНТ 32113 для 11 класса

Для заданий 1-5 требуется разработать алгоритмы на языке блок-схем, псевдокоде или естественном языке.

1. Найти сумму ряда чисел с точностью ε, $|x| \le 1$

$$1-\sqrt[4]{1-x}=\frac{x}{4}+\frac{3x^2}{4\cdot 8}+\frac{3\cdot 7x^3}{4\cdot 8\cdot 12}+\frac{3\cdot 7\cdot 11x^4}{4\cdot 8\cdot 12\cdot 16}+\frac{3\cdot 7\cdot 11\cdot 15x^5}{4\cdot 8\cdot 12\cdot 16\cdot 20}+\cdots+\frac{3\cdot 7\cdot 11\cdot\cdots\cdot(4n-5)x^n}{4\cdot 8\cdot 12\cdot 16\cdot\cdots\cdot(4n)}$$

Решение. Положим переменную *a* равной первому слагаемому $\frac{1x}{4}$, а переменную *s* равной переменной *a*. Далее в цикле переменную *a* надо умножать на рекуррентное соотношение, которое в данном случае равно $\frac{(4n-5)x}{(4n)}$, и прибавлять к переменной *s*. Цикл продолжается пока модуль очередного слагаемого не станет меньше точности ε.

```
алг СуммаРяда()
нач
  вещ x, e, a, s
  цел n

  ввод x, e
  если abs(x) > 1 или e > 1 или e <= 0 то
    вывод "Некорректные исходные данные"
  иначе

    a = x / 4
    s = a
    n = 2
    пока abs(a) >= e
    нц
      a = a * (4 * n – 3) * x / (4 * n)
      s = s + a
      n = n + 1
    кц
    вывод s

  всё
кон
```

2. Даны *n* целых чисел $a_1, a_2, ..., a_n$ и натуральное число $m > 1$. Составить алгоритм для распределения чисел $a_1, a_2, ..., a_n$ так, чтобы сначала шли (по возрастанию абсолютной величины) все числа, дающие при делении на *m* остаток 0, затем 1, 2, ..., $m - 1$.

Решение. Для решения этой задачи надо использовать любой метод сортировки, но сравнивать не сами числа, а остаток от деления числа на *m*. При равенстве остатков от деления сравнивать модули чисел.

```
алг Перестановка()
нач
  цел n, a[n], m, i

  ввод m, n
  если m <= 0 или n <= 0 то
    вывод "Некорректные исходные данные"
  иначе

  для i от 1 до n
  нц
    ввод a[i]
  кц

  QuickSortByMod(a, 1, n, m)
```

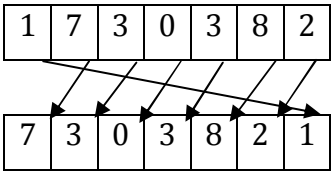
```
  для i от 1 до n
  нц
    вывод a[i]
  кц

  всё
кон

алг QuickSortByMod(арг рез вещ x[10000], арг цел n1, n2, m)
нач
  цел i, j,
  вещ y, k

  если n2 - n1 = 1 то
    если x[n1] mod m < x[n2] mod m то
      y = x[n1]
      x[n1] = x[n2]
      x[n2] = y
    всё
  иначе
    если n2 - n1 > 1 то
      k = x[(n1 + n2) div 2] mod m
      i = n1
      j = n2
      повторять
        пока (x[i] mod m > k)
        нц
          i = i + 1
        кц
        пока (x[j] mod m < k)
        нц
          j = j - 1
        кц
        если i <= j то
          y = x[i]
          x[i] = x[j]
          x[j] = y
          i = i + 1
          j = j - 1
        всё
      до i > j
      QuickSort(x, n1, j)
      QuickSort(x, i, n2)
    всё
  всё
кон
```

3. К массиву цифр натуральных чисел a ($a > 9$) применяется операция циклический **сдвиг** влево. Пример применения этой операции к числу 1730382 показан на рисунке.



Из числа 1730382 получено число 7303821. К этому числу опять можно применить сдвиг. К полученному тоже. Получается последовательность чисел 1730382, 7303821, 3038217, 0382173, 3821730, 8217303, 2173038, 1730382, ….

Составьте алгоритм, который для массива натуральных чисел a, которые могут содержать до 100 цифр, находит и выводит на экран массив наибольших чисел, получаемых сдвигами.

Решение (1 способ). Для каждого числа a сформируем массив цифр этого числа путём деления его на 10 и сохранения остатков от деления. При этом самая младшая цифра (с весом 10^0) будет записана в элемент массива с индексом 1, предпоследняя цифра (с весом 10^1) будет записана в элемент массива с индексом 2, и т.д. Также надо

подсчитать n – количество цифр числа. Само число a используем как начальное значение максимума. Далее надо $(n - 1)$ раз выполнить сдвиг массива цифр влево (после циклического сдвига на n позиций получится исходное число), получить соответствующее число и сравнить его с максимумом. Для выполнения сдвига влево надо запомнить n-ую (самую левую) цифру, далее все цифры с $(n - 1)$-ой до 1-ой записать в соседний слева элемент массива (т.е. записать i-ый элемент в $(i + 1)$-ый), а в 1-ый элемент записать запомненную цифру.

```
алг Сдвиг()
нач
  цел a, d[100], dn, n, max, i, j

  ввод a
  если a <= 9 то
    вывод "Некорректные исходные данные"
  иначе

  max = a

  n = 0
  пока a > 0
  нц
    n = n + 1
    d[n] = a mod 10
    a = a div 10
  кц

  для i от 1 до n - 1
  нц
    dn = d[n]
    для j от n - 1 до 1 шаг -1
    нц
      d[j + 1] = d[j]
    кц
    d[1] = dn

    a = d1[1]
    для j от 2 до n
    нц
      a = a * 10 + d[j]
    кц
    если a > max то
      max = a
    всё
  кц

  вывод max

  всё
кон
```

Решение (2 способ). Для каждого числа a будем искать наибольшее число, получаемое с помощью сдвига, следующим образом. Прежде всего, надо сформировать массив цифр этого числа путём деления его на 10 и сохранения остатков от деления. При этом самая младшая цифра (с весом 10^0) будет записана в элемент массива с индексом 1, предпоследняя цифра (с весом 10^1) будет записана в элемент массива с индексом 2, и т.д. Также надо подсчитать n – количество цифр числа. Далее в массиве цифр надо найти максимальную цифру числа, а также подсчитать количество вхождений максимальной цифры и запомнить позиции, в которых стоит максимальная цифра. Далее для каждого вхождения максимальной цифры сдвигаем исходное число так, чтобы максимальная цифра оказывалась самой первой (старшей), получаем соответствующее число и находим максимум из полученных чисел.

Для сдвига максимальной цифры в первую позицию надо сделать следующее. Пусть эта цифра стоит в i-ой позиции. Тогда из исходного массива цифр переписываем в новый массив сначала цифры, начиная с i-ой позиции и заканчивая 1-ой позицией,

причём цифру из *i*-ой позиции записываем в *n*-ую позицию, цифру из (*i* – 1)-ой позиции записываем в (*n* – 1)-ую позицию и т.д. Затем дописываем в новый массив цифры, начиная с *n*-ой позиции и заканчивая (*i* + 1)-ой позицией.

```
алг Сдвиг()
нач
  цел a, d[100], d1[100], n, max, dmax, i, j

  ввод a
  если a <= 9 то
    вывод "Некорректные исходные данные"
  иначе

  n = 0
  пока a > 0
  нц
    n = n + 1
    d[n] = a mod 10
    a = a div 10
  кц

  dmax = d[1]
  для i от 1 до n
  нц
    если d[i] > dmax то
      dmax = d[i]
    всё
  кц

  max = 0
  для i от 1 до n
  нц
    если d[i] = dmax то
      для j от i до 1 шаг -1
      нц
        d1[n – i + j] = d[j]
      кц
      для j от n до i + 1 шаг -1
      нц
        d1[j – i] = d[j]
      кц
      a = d1[1]
      для j от 2 до n
      нц
        a = a * 10 + d[j]
      кц
      если a > max то
        max = a
      всё
    всё
  кц

  вывод max

  всё
кон
```

4. Рассмотрим возрастающий ряд всех положительных несократимых правильных дробей, знаменатель которых меньше или равен *n*. Разработайте алгоритм нахождения суммы *P* тех членов данного ряда, для которых знаменатель нацело делится на 21.

Решение. Знаменатели дробей составляют ряд 21, 42, 63 и т.д. пока знаменатель меньше или равен *n*. Для каждого знаменателя *x* перебираем все возможные числители дроби, которые находятся в пределах от 1 до *x* – 1. Если дробь является несократимой, запоминаем её. Дробь является несократимой, если НОД числителя и знаменателя равен 1. Для нахождения НОД используем алгоритм Евклида: из большего числа надо вычитать меньшее, пока числа не станут равными.

Для ускорения работы алгоритма можно использовать не простой, а расширенный алгоритм Евклида, который заключается в следующем. Сначала надо составить два исходных уравнения.

$x \cdot u_1 + y \cdot v_1 = x$

$x \cdot u_2 + y \cdot v_2 = y$

Для того чтобы уравнения выполнялись, коэффициенты должны иметь следующие значения: $u_1 = 1$, $v_1 = 0$, $u_2 = 0$, $v_2 = 1$. После этого из первого уравнения вычитается второе, умноженное на результат целочисленного деления x на y (обозначим как q).

$x \cdot (u_1 - q \cdot u_2) + y \cdot (v_1 - q \cdot v_2) = x - q \cdot y$

В правой части уравнения получается остаток от деления x на y. На следующем шаге те же действия выполняются над вторым и третьим уравнениями. Алгоритм прекращает работу, когда правая часть уравнения становится равной 0. Значение в правой части уравнения на предпоследнем шаге даёт наибольший общий делитель чисел x и y.

Чтобы сложить запомненные дроби, надо найти НОК всех знаменателей, привести дроби к общему знаменателю, сложить числители и упростить дробь.

НОК нескольких чисел вычисляется через последовательные вычисления НОК пар чисел – НОК(НОК($a_1, a_2, \ldots, a_{n-1}$), a_n). Получить НОК двух чисел можно, воспользовавшись формулой НОК(x, y) = $\dfrac{x \cdot y}{\text{НОД}(x, y)}$.

```
алг СуммаДробей()
нач
  цел n, x, y, num[n * n], den[n * n], k, nok, p, wp, i

  ввод n
  если n <= 0 то
    вывод "Некорректные исходные данные"
  иначе

    k = 0
    для x от 21 до n шаг 21
    нц
      для y от 1 до x - 1
      нц
        если НОД(x, y) = 1 то
          k = k + 1
          num[k] = y
          den[k] = x
        всё
      кц
    кц

    nok = den[1]
    для i от 2 до k
    нц
      nok = nok * den[i] / НОД(nok, den[i])
    кц

    p = 0
    для i от 1 до k
    нц
      p = p + num[i] * nok / den[i]
    кц

    wp = p div nok
    p  = p mod nok

    вывод "Результат = "
    если wp > 0 то
      вывод wp, " "
    всё
    вывод p, " / ", nok
```

```
  всё
кон

алг НОД(арг цел x, y)
нач
  цел q, r, u1, u2, v1, v2, u, v, d

  u1 = 1
  u2 = 0
  v1 = 0
  v2 = 1

  пока y > 0
  нц
    q = x div y
    r = x mod y

    u = u1 - q * u2
    v = v1 - q * v2

    x = y
    y = r

    u1 = u2
    u2 = u
    v1 = v2
    v2 = v
  кц

  вернуть x
кон
```

5. Числа Пелля задаются соотношением:

$$P_n = \begin{cases} 0, n = 0 \\ 1, n = 1 \\ 2P_{n-1} + P_{n-2}, n > 1 \end{cases}$$

. Разработайте алгоритм, находящий простые числа Пелля в диапазоне от *P* до *Q*.

Решение. Создадим массив и проинициализируем его первые два элемента значениями 0 и 1. Далее будем вычислять следующие элементы массива – каждый *n*-ый элемент вычисляется по формуле $2P_{n-1} + P_{n-2}$. Прекращаем вычисления, когда очередной элемент массива будет больше или равен *Q*. Далее надо в полученном массиве найти простые числа.

Можно обойтись без массива. Объявляем две переменных и инициализируем их значениями 0 и 1. Далее вычисляем значение третьей переменной по такой же формуле. Пока значение третьей переменной меньше *P*, просто вычисляем эти значения. Далее пока получаемые значения меньше или равны *Q*, проверяем их на простоту. При переходе к следующему шагу цикла надо сместить значения, записав вторую переменную в первую, а третью – во вторую.

Все найденные числа надо проверить на простоту. Можно примитивно проверять делится ли число *x* на какое-либо другое число из диапазона от 2 до $\sqrt{x}$. А можно построить массив простых чисел в диапазоне от 2 до *Q* с помощью решета Эратосфена и проверять каждое число Пелля на попадание в этот массив.

Для использования решета Эратосфена необходимо построить массив чисел в заданном диапазоне от 1 до *Q*. Поскольку число 1 не является простым, в элемент массива с индексом 1 занесём значение 0 – будет удобнее, если индекс массива и число в массиве совпадают. Затем для чисел *i* в диапазоне от 2 до $\sqrt{Q}$, начиная с числа *i*, вычёркиваем из массива (заменяем нулями) все числа с шагом *i* (само число *i* не

вычёркивается). Для нахождения следующего значения *i* нужно найти первый незачёркнутый (ненулевой) элемент массива после текущего значения *i*.

```
цел nums[q]

алг ПростыеЧислаПелля()
нач
  цел p, q, p1, p2, p3

  ввод p, q
  если p <= 0 или q <= 0 или p > q то
    вывод "Некорректные исходные данные"
  иначе

    РешетоЭратосфена(q)

    p1 = 0
    p2 = 1
    p3 = 2 * p2 + p1
    пока p3 < p
    нц
      p1 = p2
      p2 = p3
      p3 = 2 * p2 + p1
    кц

    пока p3 <= q
    нц
      если nums[p3] <> 0 то
        вывод p3
      всё
      p1 = p2
      p2 = p3
      p3 = 2 * p2 + p1
    кц

  всё
кон

алг РешетоЭратосфена(арг цел n)
  цел i

  nums[1] = 0
  для i от 2 до n
  нц
    nums[i] = i
  кц

  i = 2
  пока i <= целая_часть(sqrt(n))
  нц
    для j от 2 * i до n шаг i
    нц
      nums[j] = 0
    кц
    выполнить
      i = i + 1
    до nums[i] <> 0
  кц
кон
```

ЗАДАНИЕ ПО ИНФОРМАТИКЕ
ВАРИАНТ 32101 для 10 класса

Для заданий 1-5 требуется разработать алгоритмы на языке блок-схем, псевдокоде или естественном языке.

1. Найти сумму ряда чисел с точностью ε, $|x| \le 1$

$$-1 + \frac{1}{\sqrt[4]{1-x}} = \frac{1x}{4} + \frac{1 \cdot 5x^2}{4 \cdot 8} + \frac{1 \cdot 5 \cdot 9x^3}{4 \cdot 8 \cdot 12} + \frac{1 \cdot 5 \cdot 9 \cdot 13x^4}{4 \cdot 8 \cdot 12 \cdot 16} + \cdots + \frac{1 \cdot 5 \cdot 9 \cdot 13 \cdot \cdots \cdot (4n-3)x^n}{4 \cdot 8 \cdot 12 \cdot 16 \cdot \cdots \cdot (4n)}$$

Решение. Положим переменную *a* равной первому слагаемому $\frac{1x}{4}$, а переменную *s* равной переменной *a*. Далее в цикле переменную *a* надо умножать на рекуррентное соотношение, которое в данном случае равно $\frac{(4n-3)x}{(4n)}$, и прибавлять к переменной *s*. Цикл продолжается пока модуль очередного слагаемого не станет меньше точности ε.

```
алг СуммаРяда()
нач
  вещ x, e, a, s
  цел n

  ввод x, e
  если abs(x) > 1 или e > 1 или e <= 0 то
    вывод "Некорректные исходные данные"
  иначе

    a = x / 4
    s = a
    n = 2
    пока abs(a) >= e
    нц
      a = a * (4 * n - 3) * x / (4 * n)
      s = s + a
      n = n + 1
    кц
    вывод s

  всё
кон
```

2. Автоморфным называют число, десятичная запись квадрата которого оканчивается цифрами самого этого числа. Найти произведение чётных автоморфных чисел с порядковыми номерами от *P* до *Q*. Среди таких чисел найти все дружественные числа. Два числа называются дружественными, если каждое из них равно сумме всех делителей другого, кроме самого этого числа.

Решение. Перебираем натуральные числа, пока не найдём автоморфное число с номером *P* – 1. Далее перебираем следующие натуральные числа, пока не найдём автоморфное число с номером *Q*. При этом запоминаем чётные автоморфные числа, а также считаем их произведение. Далее надо найти дружественные числа. Для каждого числа *x* из массива найденных чётных автоморфных чисел подсчитаем сумму его делителей. Для этого нужно проверить, на какие числа из диапазона от 2 до *x* / 2 делится число *x*, и сложить эти числа, а также прибавить 1 (т.к. любое число делится на 1). Далее перебираем все пары чисел *x* и *y* из сформированного массива, и если *x* равно сумме делителей *y*, а *y* равно сумме делителей *x*, то выводим числа *x* и *y*.

Для проверки на автоморфность надо возвести число в квадрат и сравнить последние цифры нового числа с исходным. Для этого надо взять остаток от деления нового числа на 10 в некоторой степени. Степень зависит от количества цифр в исходном числе. Поэтому исходное число делим несколько раз на 10, пока не получится 0,

одновременно столько же раз умножаем на 10 делитель, изначально равный 1. Затем берём остаток от деления нового числа на полученный делитель, и если этот остаток от деления равен исходному числу, то исходное число является автоморфным.

Для того чтобы не вычислять делитель для каждого числа можно поступить следующим образом. Сначала положим делитель равным 10. Когда проверяемое число станет равным делителю, умножаем делитель на 10.

```
алг АвтоморфныеЧисла()
нач
  цел p, q, n, i, j, k, d, amn[q], sum[q]

  ввод x, e
  если p <= 0 или q <= 0 или u <= 0 или p > q то
    вывод "Некорректные исходные данные"
  иначе

    n = 0
    i = 0
    d = 10
    пока i <= p - 1
    нц
      если n ^ 3 mod d = n то
        i = i + 1
      всё
      n = n + 1
      если n = d то
        d = d * 10
      всё
    кц

    k = 0
    пока i <= q
    нц
      если n ^ 3 mod d = n то
        i = i + 1
        если n mod 2 = 0 то
          k = k + 1
          amn[k] = n
          sum[k] = СуммаДелителей(n)
        всё
      всё
      n = n + 1
      если n = d то
        d = d * 10
      всё
    кц

    для i от 1 до k - 1
    нц
      для j от i + 1 до k
      нц
        если sum[i] = sum[j] то
          вывод amn[i], amn[j]
        всё
      кц
    кц

  всё
кон

алг СуммаДелителей(арг цел n)
нач
  цел s, i

  s = 1
  для i от 2 до n div 2
  нц
    если n mod i = 0 то
      s = s + i
    всё
  кц
  вернуть s
кон
```

3. На координатной плоскости по линиям сетки построено несколько прямоугольников. Необходимо подсчитать число точек с целочисленными координатами, принадлежащими сразу всем этим прямоугольникам.

Решение. Будем считать, что прямоугольник задаётся четырьмя числами – координаты *x* и *y* левого нижнего угла и координаты *x* и *y* правого верхнего угла. Числа должны быть целыми. Можно использовать четыре одномерных массива. Найдём прямоугольник, являющийся пересечением всех исходных прямоугольников. Для этого положим искомый прямоугольник равным первому, и будем последовательно искать пересечение искомого прямоугольника со 2-ым, 3-им и т.д. Для нахождения пересечения двух прямоугольников надо сделать следующее:

1. запоминаем длины сторон прямоугольников;
2. из левых координат двух прямоугольников выбираем минимальную и записываем в переменную x_{min}, а другую координату сохраняем как левую координату прямоугольника, являющегося пересечением;
3. из правых координат двух прямоугольников выбираем максимальную и записываем в переменную x_{max}, а другую координату сохраняем как правую координату прямоугольника, являющегося пересечением;
4. если $x_{max} - x_{min}$ больше, чем сумма длин сторон по оси *x*, значит, прямоугольники не пересекаются;
5. из нижних координат двух прямоугольников выбираем минимальную и записываем в переменную y_{min}, а другую координату сохраняем как нижнюю координату прямоугольника, являющегося пересечением;
6. из верхних координат двух прямоугольников выбираем максимальную и записываем в переменную y_{max}, а другую координату сохраняем как верхнюю координату прямоугольника, являющегося пересечением;
7. если $y_{max} - y_{min}$ больше, чем сумма длин сторон по оси *y*, значит, прямоугольники не пересекаются.

Если прямоугольники не пересекаются, можно прекратить перебор прямоугольников. После нахождения пересечения всех прямоугольников количество точек с целочисленными координатами вычисляется через координаты прямоугольника, являющегося пересечением, по формуле $(x_{прав} - x_{лев} + 1) \cdot (y_{ниж} - y_{верх} + 1)$.

```
алг ПересечениеПрямоугольников()
нач
  цел n, xl[n], xp[n], yn[n], yv[n], rxl, rxp, ryn, ryv, xmin, xmax, ymin, ymax, a, b, ar, br, i
  лог intersec

  ввод n
  если n <= 0 то
    вывод "Некорректные исходные данные"
  иначе

    для i от 1 до n
    нц
      ввод xl[i], xp[i], yn[i], yv[i]
    кц

    rxl = xl[1]
    rxp = xp[1]
    ryn = yn[1]
    ryv = yv[1]
    i = 2
    intersec = истина
    пока i <= n и intersec
    нц
      a  = xp[i] - xl[i]
      b  = yv[i] - yn[i]
      ar = rxp - rxl
```

```
      br = ryv - ryn
      если xl[i] < rxl то
        xmin = xl[i]
      иначе
        xmin = rxl
        rxl  = xl[i]
      всё
      если xp[i] > rxp то
        xmax = xp[i]
      иначе
        xmax = rxp
        rxp  = xp[i]
      всё
      если xmax - xmin > a + ra то
        intersec = ложь
      всё
      если yn[i] < ryn то
        ymin = yn[i]
      иначе
        ymin = ryn
        ryn  = yn[i]
      всё
      если yv[i] > ryv то
        ymax = yv[i]
      иначе
        ymax = ryv
        ryv  = yv[i]
      всё
      если ymax - ymin > b + rb то
        intersec = ложь
      всё
      i = i + 1
    кц

    если intersec то
      вывод "Количество точек = ", (rxp - rxl + 1) * (ryv - ryn + 1)
    иначе
      вывод "Прямоугольники не пересекаются"
    всё

  всё
кон
```

4. Рассмотрим возрастающий ряд всех положительных несократимых правильных дробей, знаменатель которых меньше или равен n. Разработайте алгоритм нахождения суммы P тех членов данного ряда, для которых знаменатель является совершенным числом. Число называется совершенным, если оно равно сумме всех своих делителей, исключая само это число.

Решение. Перебираем все числа x в диапазоне от 2 до n. Для каждого числа x надо проверить, является ли оно совершенным. Для этого рассмотрим возможные делители i в диапазоне от 2 до $x / 2$ (любое число делится на 1, поэтому изначально сумма делителей должна быть равна 1, а последний возможный делитель числа x, не равный x, есть $x / 2$), и если x делится на i, то прибавляем i к сумме делителей. Если сумма делителей равна самому числу, значит, число является совершенным. В этом случае перебираем все возможные числители дроби, которые находятся в пределах от 1 до $x - 1$. Если дробь является несократимой, запоминаем её. Дробь является несократимой, если НОД числителя и знаменателя равен 1. Для нахождения НОД используем алгоритм Евклида: из большего числа надо вычитать меньшее, пока числа не станут равными.

Для ускорения работы алгоритма можно использовать не простой, а расширенный алгоритм Евклида, который заключается в следующем. Сначала надо составить два исходных уравнения.

$x \cdot u_1 + y \cdot v_1 = x$

$x \cdot u_2 + y \cdot v_2 = y$

Для того чтобы уравнения выполнялись, коэффициенты должны иметь следующие значения: $u_1 = 1$, $v_1 = 0$, $u_2 = 0$, $v_2 = 1$. После этого из первого уравнения вычитается второе, умноженное на результат целочисленного деления x на y (обозначим как q).

$x \cdot (u_1 - q \cdot u_2) + y \cdot (v_1 - q \cdot v_2) = x - q \cdot y$

В правой части уравнения получается остаток от деления x на y. На следующем шаге те же действия выполняются над вторым и третьим уравнениями. Алгоритм прекращает работу, когда правая часть уравнения становится равной 0. Значение в правой части уравнения на предпоследнем шаге даёт наибольший общий делитель чисел x и y.

Чтобы сложить запомненные дроби, надо найти НОК всех знаменателей, привести дроби к общему знаменателю, сложить числители и упростить дробь.

НОК нескольких чисел вычисляется через последовательные вычисления НОК пар чисел – НОК(НОК($a_1, a_2, \ldots, a_{n-1}$), a_n). Получить НОК двух чисел можно, воспользовавшись формулой НОК(x, y) = $\frac{x \cdot y}{\text{НОД}(x, y)}$.

```
алг СуммаДробей()
нач
  цел n, x, y, num[n * n], den[n * n], k, nok, p, wp, i

  ввод n
  если n <= 0 то
    вывод "Некорректные исходные данные"
  иначе

    k = 0
    для x от 2 до n
    нц
      если x = СуммаДелителей(x) то
        для y от 1 до x - 1
        нц
          если НОД(x, y) = 1 то
            k = k + 1
            num[k] = y
            den[k] = x
          всё
        кц
      всё
    кц

    nok = den[1]
    для i от 2 до k
    нц
      nok = nok * den[i] / НОД(nok, den[i])
    кц

    p = 0
    для i от 1 до k
    нц
      p = p + num[i] * nok / den[i]
    кц

    wp = p div nok
    p  = p mod nok

    вывод "Результат = "
    если wp > 0 то
      вывод wp, " "
    всё
    вывод p, " / ", nok

  всё
кон

алг СуммаДелителей(арг цел n)
нач
```

```
  цел s, i

  s = 1
  для i от 2 до n div 2
  нц
    если n mod i = 0 то
      s = s + i
    всё
  кц
  вернуть s
кон

алг НОД(арг цел x, y)
нач
  цел q, r, u1, u2, v1, v2, u, v, d

  u1 = 1
  u2 = 0
  v1 = 0
  v2 = 1

  пока y > 0
  нц
    q = x div y
    r = x mod y

    u = u1 - q * u2
    v = v1 - q * v2

    x = y
    y = r

    u1 = u2
    u2 = u
    v1 = v2
    v2 = v
  кц

  вернуть x
кон
```

5. Сопутствующие числа Пелля задаются соотношением:

$$P_n = \begin{cases} 2, n = 0 \\ 2, n = 1 \\ 2P_{n-1} + P_{n-2}, n > 1 \end{cases}$$

. Разработайте алгоритм, находящий сопутствующие числа Пелля в диапазоне от P до Q, которые являются простыми.

Решение. Создадим массив и проинициализируем его первые два элемента значением 2. Далее будем вычислять следующие элементы массива – каждый n-ый элемент вычисляется по формуле $2P_{n-1} + P_{n-2}$. Прекращаем вычисления, когда очередной элемент массива будет больше или равен Q. Далее надо в полученном массиве найти простые числа.

Можно обойтись без массива. Объявляем две переменных и инициализируем их значением 2. Далее вычисляем значение третьей переменной по такой же формуле. Пока значение третьей переменной меньше P, просто вычисляем эти значения. Далее пока получаемые значения меньше или равны Q, проверяем их на простоту. При переходе к следующему шагу цикла надо сместить значения, записав вторую переменную в первую, а третью – во вторую.

Все найденные числа надо проверить на простоту. Можно примитивно проверять делится ли число x на какое-либо другое число из диапазона от 2 до $\sqrt{x}$. А можно построить массив простых чисел в диапазоне от 2 до Q с помощью решета Эратосфена и проверять каждое число Пелля на попадание в этот массив.

Для использования решета Эратосфена необходимо построить массив чисел в заданном диапазоне от 1 до Q. Поскольку число 1 не является простым, в элемент массива с индексом 1 занесём значение 0 – будет удобнее, если индекс массива и число в массиве совпадают. Затем для чисел i в диапазоне от 2 до $\sqrt{Q}$, начиная с числа i, вычёркиваем из массива (заменяем нулями) все числа с шагом i (само число i не вычёркивается). Для нахождения следующего значения i нужно найти первый незачёркнутый (ненулевой) элемент массива после текущего значения i.

```
цел nums[q]

алг ПростыеЧислаПелля()
нач
  цел p, q, p1, p2, p3

  ввод p, q
  если p <= 0 или q <= 0 или p > q то
    вывод "Некорректные исходные данные"
  иначе

    РешетоЭратосфена(q)

    p1 = 2
    p2 = 2
    p3 = 2 * p2 + p1
    пока p3 < p
    нц
      p1 = p2
      p2 = p3
      p3 = 2 * p2 + p1
    кц

    пока p3 <= q
    нц
      если nums[p3] <> 0 то
        вывод p3
      всё
      p1 = p2
      p2 = p3
      p3 = 2 * p2 + p1
    кц

  всё
кон

алг РешетоЭратосфена(арг цел n)
  цел i

  nums[1] = 0
  для i от 2 до n
  нц
    nums[i] = i
  кц

  i = 2
  пока i <= целая_часть(sqrt(n))
  нц
    для j от 2 * i до n шаг i
    нц
      nums[j] = 0
    кц
    выполнить
      i = i + 1
    до nums[i] <> 0
  кц
кон
```

ЗАДАНИЕ ПО ИНФОРМАТИКЕ
ВАРИАНТ 31991 для 9 класса

Для заданий 1-5 требуется разработать алгоритмы на языке блок-схем, псевдокоде или естественном языке.

1. В теории чисел нечётное натуральное число *k* называют числом Серпинского, если для любого натурального числа *n* число $k \times 2^n + 1$ является составным. Разработайте алгоритм поиска чисел Серпинского для *k* = U в диапазоне для *n* от P до Q.

Решение. Необходимо перебрать все числа *n* в диапазоне от P до Q и для каждого *n* проверить, что число $s = k \times 2^n + 1$ является составным. Для этого необходимо проверить, делится ли число *s* на какое-то другое число в диапазоне от 3 до $\sqrt{s}$ (на 2 не делится, т.к. 2^n – чётное число, следовательно, $k \times 2^n + 1$ – нечётное число). Для упрощения проверки можно заранее построить массив простых чисел в диапазоне от 2 до Q с помощью решета Эратосфена.

Для использования решета Эратосфена необходимо построить массив чисел в заданном диапазоне от 1 до Q. Поскольку число 1 не является простым, в элемент массива с индексом 1 занесём значение 0 – будет удобнее, если индекс массива и число в массиве совпадают. Затем для чисел *i* в диапазоне от 2 до $\sqrt{Q}$, начиная с числа *i*, вычёркиваем из массива (заменяем нулями) все числа с шагом *i* (само число *i* не вычёркивается). Для нахождения следующего значения *i* нужно найти первый незачёркнутый (ненулевой) элемент массива после текущего значения *i*.

```
цел nums[1000000]

алг ЧислаСерпинского()
нач
  цел p, q, k, u, n, s

  ввод p, q, u
  если p <= 0 или q <= 0 или u <= 0 или p > q то
    вывод "Некорректные исходные данные"
  иначе

    k = u
    s = k * 2 ^ q + 1

    РешетоЭратосфена(s)

    для n от p до q
    нц
      s = k * 2 ^ n + 1
      если nums[s] = 0 то
        вывод s
      всё
    кц

  всё
кон

алг РешетоЭратосфена(арг цел n)
  цел i

  nums[1] = 0
  для i от 2 до n
  нц
    nums[i] = i
  кц

  i = 2
  пока i <= целая_часть(sqrt(n))
  нц
    для j от 2 * i до n шаг i
```

```
    нц
      nums[j] = 0
    кц
    выполнить
      i = i + 1
    до nums[i] <> 0
  кц
кон
```

2. Известно, что десятизначное число A = 2013*x*2013*y* делится нацело на 121. Составьте алгоритм для нахождения всех возможных пар цифр (*x*, *y*).

Решение – 1 способ. Перебираем все *x* в диапазоне от 0 до 9 и для каждого *x* перебираем все *y* в диапазоне от 0 до 9. Для всех возможных пар проверяем остаток от деления на 121.

Решение – 2 способ. Чтобы число делилось на 121, оно должно дважды делится на 11. На 11 число делится, если разность между суммой цифр в чётных позициях и суммой цифр в нечётных позициях делится на 11. Для числа 2013*x*2013*y* эта разность равна (2 + 1 + *x* + 0 + 3) – (0 + 3 + 2 + 1 + *y*) = *x* – *y*. Поскольку *x* и *y* – цифры, *x* – *y* может делится на 11, только если *x* = *y*. Поэтому можно перебрать все *x* в диапазоне от 0 до 9, для каждого *x* вычисляем результат деления числа 2013*x*2013*x* на 11 и проверить, делится ли этот результат на 11.

Решение – 3 способ. Чтобы число делилось на 121, оно должно дважды делится на 11. На 11 число делится, если разность между суммой цифр в чётных позициях и суммой цифр в нечётных позициях делится на 11. Для числа 2013*x*2013*y* эта разность равна (2 + 1 + *x* + 0 + 3) – (0 + 3 + 2 + 1 + *y*) = *x* – *y*. Поскольку *x* и *y* – цифры, *x* – *y* может делится на 11, только если *x* = *y*. Представим число 2013*x*2013*x* в следующем виде $2013 \cdot 10^6 + x \cdot 10^5 + 2013 \cdot 10 + x$. 2013 / 11 = 183, значит, $2013 \cdot 10^6 + 2013 \cdot 10$ / 11 = $183 \cdot 10^6 + 183 \cdot 10$. Поскольку в этом числе одинаковые цифры оказываются в разных (чётных и нечётных) позициях, разность между их суммами равно 0, и число делится на 11. Значит, $2013 \cdot 10^6 + 2013 \cdot 10$ делится на 121. Следовательно, необходимо чтобы число $x \cdot (10^5 + 1)$ тоже делилось на 121. $10^5 + 1$ делится на 11, результат равен 9091. Но это число не делится на 11. Поэтому $x \cdot (10^5 + 1)$ может делиться на 121 только при $x = 0$.

3. Разработайте алгоритм для решения задачи: найти все натуральные числа, не превосходящие заданного числа N и делящиеся нацело на куб каждой из своих цифр.

Решение. Перебираем все числа *i* в диапазоне от 1 до N. Для каждого числа *i* надо найти его цифры (для чего последовательно нужно делить его на 10) и проверить остаток от деления *i* на куб каждой из его цифр.

```
алг ДелениеНаКубЦифр()
нач
  цел n, i, k, d
  лог f

  ввод n

  если n <= 0 то
    вывод "Некорректные исходные данные"
  иначе

    для i от 1 до n
    нц
      k = i
      f = истина
      пока k > 0 и f
```

```
      нц
        d = k mod 10
        k = k div 10
        если i mod d <> 0 то
          f = ложь
        всё
      кц
      если f то
        вывод i
      всё
    кц
  всё
кон
```

4. Даны три стопки карточек, на каждой из которых записано два числа. Эти два числа задают координаты точки на плоскости. Посчитать число треугольников общего вида и прямоугольных треугольников среди троек.

Решение. Для каждой тройки точек (x_1, y_1), (x_2, y_2) и (x_3, y_3) надо проверить, можно ли построить треугольник. Для этого надо найти три расстояния между парами точек по формуле $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ и проверить, что сумма каждых двух расстояний больше третьего. Чтобы треугольник был прямоугольным, скалярное произведение каких-то двух векторов, образованных сторонами треугольника, должно быть равно 0, т.е., например, $(x_3 - x_1) \cdot (x_2 - x_1) + (y_3 - y_1) \cdot (y_2 - y_1)$ должно быть равно 0.

```
алг Треугольники()
нач
  цел n, x1[n], y1[n], x2[n], y2[n], x3[n], y3[n], r1, r2, r3, i, t, rt
  лог f

  ввод n

  если n <= 0 то
    вывод "Некорректные исходные данные"
  иначе

    t = 0
    rt = 0
    для i от 1 до n
    нц
      r1 = sqrt(sqr(x1[i] - x2[i]) + sqr(y1[i] - y2[i]))
      r2 = sqrt(sqr(x1[i] - x3[i]) + sqr(y1[i] - y3[i]))
      r3 = sqrt(sqr(x2[i] - x3[i]) + sqr(y2[i] - y3[i]))
      если r1 + r2 > r3 и r1 + r3 > r2 и r2 + r3 > r1 то
        t = t + 1
        если (x3[i] - x1[i]) * (x2[i] - x1[i]) + (y3[i] - y1[i]) * (y2[i] - y1[i]) = 0 или
             (x1[i] - x2[i]) * (x3[i] - x2[i]) + (y1[i] - y2[i]) * (y3[i] - y2[i]) = 0 или
             (x1[i] - x3[i]) * (x2[i] - x3[i]) + (y1[i] - y3[i]) * (y2[i] - y3[i]) = 0 то
          rt = rt + 1
        всё
      всё
    кц
    вывод "Количество треугольников общего вида   - ", t
    вывод "Количество прямоугольных треугольников - ", rt

  всё
кон
```

5. Найти количество пар взаимно простых чисел из диапазона от P до Q, в которых хотя бы одно число из пары является совершенным.

Решение. Перебираем все возможные пары чисел из диапазона от P до Q. Для каждой пары проверяем, что числа являются взаимно простыми, т.е. их наибольший общий делитель равен 1. Если это так, то надо проверить, что хотя бы одно число является совершенным. Для проверки того, что число n является совершенным, надо

последовательно проверять числа в диапазоне от 1 до n / 2 и суммировать те, которые являются делителями числа n. Если сумма делителей равна n, значит, число n является совершенным.

Для поиска НОД можно использовать простой или расширенный алгоритм Евклида. Простой алгоритм Евклида заключается в вычитании меньшего числа из большего, пока числа не станут равными. Расширенный алгоритм заключается в следующем. Сначала надо составить два исходных уравнения.

$x \cdot u_1 + y \cdot v_1 = x$

$x \cdot u_2 + y \cdot v_2 = y$

Для того чтобы уравнения выполнялись, коэффициенты должны иметь следующие значения: $u_1 = 1$, $v_1 = 0$, $u_2 = 0$, $v_2 = 1$. После этого из первого уравнения вычитается второе, умноженное на результат целочисленного деления x на y (обозначим как q).

$x \cdot (u_1 - q \cdot u_2) + y \cdot (v_1 - q \cdot v_2) = x - q \cdot y$

В правой части уравнения получается остаток от деления x на y. На следующем шаге те же действия выполняются над вторым и третьим уравнениями. Алгоритм прекращает работу, когда правая часть уравнения становится равной 0. Значение в правой части уравнения на предпоследнем шаге даёт наибольший общий делитель чисел x и y.

```
алг ПарыЧисел()
нач
  цел p, q, m, n, k

  ввод p, q, u
  если p <= 0 или q <= 0 или u <= 0 или p > q то
    вывод "Некорректные исходные данные"
  иначе

    k = 0
    для m от p до q - 1
    нц
      для n от m + 1 до q
      нц
        если НОД(m, n) = 1 и (СовершенноеЧисло(n) или СовершенноеЧисло(m)) то
          k = k + 1
        всё
      кц
    кц

  всё
кон

алг НОД(арг цел x, y)
нач
  цел q, r, u1, u2, v1, v2, u, v, d

  u1 = 1
  u2 = 0
  v1 = 0
  v2 = 1

  пока y > 0
  нц
    q = x div y
    r = x mod y

    u = u1 - q * u2
    v = v1 - q * v2

    x = y
    y = r

    u1 = u2
```

```
    u2 = u
    v1 = v2
    v2 = v
  кц

  вернуть x
кон

алг СовершенноеЧисло(арг цел n)
нач
  цел s, i

  s = 1
  для i от 2 до n div 2
  нц
    если n mod i = 0 то
      s = s + i
    всё
  кц
  если s = n то
    вернуть истина
  иначе
    вернуть ложь
  всё
кон
```