# Материалы заданий отборочного этапа Олимпиады школьников «Надежда энергетики» по предмету «информатика» в 2016/2017 учебном году.

1. Тау-число – это целое число $n$, делящееся на число своих делителей. Положительное число N свободно от квадратов тогда и только тогда, когда в разложении этого числа на простые множители ни одно простое число не встречается больше одного раза. Разработайте алгоритм поиска свободных от квадратов чисел в диапазоне от P до Q, являющихся тау-числами. (10-11 классы)

**Решение.** Нам надо подсчитать число делителей и проверить, делится ли число на квадрат какого-либо другого числа. Совместим эти проверки в одном цикле. Для всех чисел $n$ в диапазоне от P до Q рассмотрим возможные делители $i$ в диапазоне от 2 до $n$ / 2 (любое число делится на 1, поэтому изначально количество делителей должно быть равно 1, кроме того, любое число делится на себя, поэтому к количеству делителей нужно прибавить ещё 1, а последний возможный делитель числа $n$, не равный $n$, есть $n$ / 2). Если $n$ делится на $i$, то увеличиваем количество делителей, а если $n$ делится на $i^2$, то увеличиваем количество квадратов, на которые делится $n$. Чтобы число $n$ удовлетворяло требованиям задачи, надо проверить, что $n$ делится на количество делителей, а количество квадратов равно 0.

Можно разделить цикл на два – от 2 до $\sqrt{n}$ и от $\sqrt{n}+1$ до $n$ / 2. В первом цикле делаются обе проверки, а во втором проверяем только, является число делителем $n$.

```
алг ТауЧислоСвободноеОтКвадратов
нач
  цел p, q, n, i, f
  лог free

  ввод p, q

  для n от p до q
  нц
    f = 2
    free = истина
    i = 2
    пока i <= целая_часть(sqrt(n)) и free
    нц
      если n mod i = 0 то
        f = f + 1
      всё
      если n mod (i * i) = 0 то
        free = ложь
      всё
      i = i + 1
    кц
    если free то
      для i от целая_часть(sqrt(n)) + 1 до n div 2
      нц
        если n mod i = 0 то
          f = f + 1
        всё
      кц
    всё
    если n mod f = 0 и free то
      вывод n
    всё
  кц
кон
```

2. Собралась Алёнушка с братцем Иванушкой в гости к тётушке в соседнюю деревню. Дорога шла лесом. Прошёл дождь, и на дороге образовалось целых N луж. А Баба Яга уж очень завидовала сестрице Алёнушке и, узнав, что та собирается в путь, решила в очередной раз досадить ей: сварила колдовское месиво и наделала из него ядовитых шариков. Пролетела на помеле над лесной дорогой и раскидала случайным образом свои шарики по всем лужам: где промазала, где 1, где 2, где 3 шарика закинула. Растворились они в луже. А кто наступит на лужу с одним шариком – через час ягнёночком станет, с двумя – козлёночком, а с тремя – щенком беспородным. Пошли той дорогой Алёнушка с братцем, Алёнушка аккуратно идёт и братцу наказывает – «Не мочи ножки, заболеешь». А тот бежит по

дорожке, где перепрыгнет, где обойдёт лужу. Три раза не удержался и намочил ножки в разных лужах. Определите – в каком обличье придёт Иванушка в гости к тётушке? (9-11 классы)

**Решение.** Смоделируем поведение Бабы Яги и случайно накидаем некоторое количество ядовитых шариков в каждую из N луж. Потом смоделируем поведение Иванушки и наступим в три лужи. Проверим, сколько в каждой луже растворилось шариков. Максимальное значение и определит облик Иванушки.

```
алг Иванушка
нач
  цел puddles[n], n, res, i, k

  ввод n
  для i от 1 до n
  нц
    puddles[i] = случайное(0, 3)                      // Функция случайное генерирует случайное целое число,
  кц                                                  // входящее в указанный диапазон

  res = 0
  для i от 1 до 3
  нц
    k = случайное(1, n)
    если puddles[k] > res то
      res = puddles[k]
    всё
  кц

  если res = 3 то
    вывод "В этот раз Иванушка превратится в щенка беспородного"
  иначе
    если res = 2 то
      вывод "В этот раз Иванушка превратится в козлёночка"
    иначе
      если res = 1 то
        вывод "В этот раз Иванушка превратится в ягнёночка"
      иначе
        вывод "В этот раз Иванушке повезло, и он останется человеком"
      всё
    всё
  всё
кон
```

**3.** В теории чисел безопасное простое число – это простое число вида $2p + 1$, где $p$ также простое. Вам предлагается разработать алгоритм для нахождения безопасных простых чисел в диапазоне от F до G. (10-11 классы)

**Решение.** Построим массив простых чисел в диапазоне от 2 до G с помощью решета Эратосфена. Далее возьмём из построенного массива простые числа $p$ в диапазоне от F / 2 до (G – 1) / 2 и проверим с помощью того же массива, является число $2p + 1$ также простым.

Для использования решета Эратосфена необходимо построить массив чисел в заданном диапазоне от 1 до G. 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 $\sqrt{G}$, 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 $i$.

```
алг БезопасныеПростыеЧисла
нач
  цел primes[g], f, g, p, i, j

  ввод f, g

  // Построение массива простых чисел
  primes[1] = 0
  для i от 2 до g
  нц
    primes[i] = i
  кц
```

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

  // Поиск безопасных простых чисел
  для p от f div 2 до (g – 1) div 2
  нц
    если primes[p] <> 0 и primes[2 * p + 1] <> 0 то
      вывод 2 * p + 1, " является безопасным простым числом"
    всё
  кц

кон
```

**4.** В математике свободным от квадратов, или бесквадратным, называется число, которое не делится ни на один квадрат, кроме 1. Разработайте алгоритм нахождения простых свободных от квадратов чисел в диапазоне от *m* до *n*. (9 класс)

**Решение.** Можно найти разложение числа на простые сомножители и проверить сколько раз каждый сомножитель встречается в разложении. Но для этого надо строить массив простых чисел или проверять каждое число на простоту, что достаточно трудоёмко. Проще для каждого числа *i* в диапазоне от *m* до *n* просмотреть все числа от 2 до $\sqrt{i}$ и проверить, делится ли *i* на квадрат какого-нибудь из них. Далее проверить является ли число *i* простым.

Но вообще-то любое простое число будет свободным от квадратов, т.к. оно не делится вообще ни на что. Так что можно просто проверить является ли число простым или построить массив простых чисел с помощью решета Эратосфена и выбрать из него числа, входящие в диапазон от *m* до *n*.

Для использования решета Эратосфена необходимо построить массив чисел в заданном диапазоне от 1 до *n*. 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 $\sqrt{n}$, 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*.

**5.** Простое число Ньюмена-Шэнкса-Уильямса (NSW-простое) – это простое число, которое можно записать в виде:

$$S_0 = 1,\quad S_1 = 1,\quad S_n = 2S_{n-1} + S_{n-2}, n \ge 2$$

Вам предлагается разработать алгоритм для нахождения простых чисел указанного вида в диапазоне от F до G. (9-11 класс)

**Решение.** Создадим массив и проинициализируем его первые два элемента единицами. Далее будем вычислять следующие элементы массива – каждый *n*-ый элемент вычисляется по формуле $2S_{n-1} + S_{n-2}$. Прекращаем вычисления, когда очередной элемент массива будет больше или равен G. Затем выводим те элементы массива, которые попадают в диапазон от F до G.

Можно обойтись без массива. Объявляем две переменных и инициализируем их единицами. Если F меньше или равно 1, надо вывести эти значения. Далее вычисляем значение третьей переменной по такой же формуле. Пока значение третьей переменной меньше F, просто вычисляем эти значения. Далее пока получаемые значения меньше или равны G, выводим их.

При переходе к следующему шагу цикла надо сместить значения, записав вторую переменную в первую, а третью – во вторую.

```
алг ПростыеЧислаНьюменаШэнксаУильямса
нач
  цел f, g, s1, s2, s3

  ввод f, g

  s1 = 1
  s2 = 1
  если f <= 1 то
    вывод s1
    вывод s2
  всё

  s3 = 2 * s2 + s1
  пока s3 < f
  нц
    s1 = s2
    s2 = s3
    s3 = 2 * s2 + s1
  кц
  пока s3 <= g
  нц
    вывод s3
    s1 = s2
    s2 = s3
    s3 = 2 * s2 + s1
  кц
кон
```

**6.** Последовательность Фарея порядка $n$ представляет собой возрастающий ряд всех положительных несократимых правильных дробей, знаменатель которых меньше или равен $n$. Разработайте алгоритм нахождения суммы P членов данного ряда. (9 класс)

**Решение.** Для всех знаменателей $m$ от 2 до $n$ перебираем все возможные числители $k$ от 1 до $m - 1$. Для каждой пары числителей и знаменателей нужно найти НОД (для этого можно использовать алгоритм Евклида – из большего числа надо вычитать меньшее, пока числа не станут одинаковы). Если НОД равен 1, значит, дробь нельзя сократить, и нужно сохранить её. Кроме того, надо сохранить дроби $\frac{0}{1}$ и $\frac{1}{1}$.

Далее надо вычислить сумму найденных дробей. Можно пойти примитивным путём и использовать вещественное деление. Но можно также найти НОК чисел от 2 до $n$, привести дроби к общему знаменателю, вычислить сумму числителей и сократить дробь. НОК нескольких чисел вычисляется через последовательные вычисления НОК пар чисел – НОК(НОК($a_1$, $a_2$, …, $a_{n-1}$), $a_n$). Получить НОК двух чисел можно, разложив их на простые сомножители или воспользовавшись формулой НОК($x$, $y$) = $\frac{x \cdot y}{\text{НОД}(x, y)}$.

```
алг СуммаПоследовательностейФарея
нач
  цел nums[10000], dens[10000], fnums[10000], fdens[10000]
  цел p, n, count, rnum, rden

  ввод p

  для n от 1 до p
  нц
    ПоследовательностьФарея(n, fnums, fdens, count)           // Вычисляем i-ую последовательность Фарея
    СуммаДробей(fnums, fdens, count, nums[n], dens[n])
  кц

  СуммаДробей(nums, dens, p, rnum, rden)
  вывод rnum div rden, " ", rnum mod rden, "/", rden
кон

алг ПоследовательностьФарея(арг цел n, рез цел fnums[10000], fdens[10000], count)
нач
  цел k, m
```

```
  fnums[1] = 0
  fdens[1] = 1
  count = 1

  для m от 2 до n
  нц
    для k от 1 до m – 1
    нц
      если НОД(k, m) = 1 то
        count = count + 1
        fnums[count] = k
        fdens[count] = m
      всё
    кц
  кц
  count = count + 1
  fnums[count] = 1
  fdens[count] = 1
кон

алг СуммаДробей(арг цел nums[10000], dens[10000], count, рез цел rnum, rden)
нач
  цел i, nod

  rden = НОК(dens, count)
  rnum = 0

  для i от 1 до count
  нц
    rnum = rnum + (rden div dens[i]) * nums[i]
  кц

  nod = НОД(rnum, rden)
  rnum = rnum div nod
  rden = rden div nod
кон

алг НОК(арг цел mas[10000], count)
нач
  цел nok, i

  nok = mas[1]
  для i от 2 до count
  нц
    nok = nok * mas[i] div НОД(nok, mas[i])
  кц
  вернуть nok
кон

алг НОД(арг цел x, y)
нач
  пока x <> y
  нц
    если x > y то
      x = x - y
    иначе
      y = y - x
    всё
  кц
  вернуть x
кон
```

Дроби из ряда Фарея можно вычислить не перебором, а по формулам. Первые две дроби ряда – это $\frac{0}{1}$ и $\frac{1}{n}$. Далее если мы знаем две дроби $\frac{a}{b}$ и $\frac{c}{d}$, то числитель следующей дроби равен $\left\lfloor \frac{n+b}{d} \right\rfloor \cdot c - a$, а знаменатель – $\left\lfloor \frac{n+b}{d} \right\rfloor \cdot d - b$ (скобки означают то, что мы используем целочисленное деление). Вычисления необходимо продолжать, пока не получится дробь $\frac{1}{1}$.

```
алг ПоследовательностьФарея(арг цел n, рез цел fnums[10000], fdens[10000], count)
нач

  fnums[1] = 0
  fdens[1] = 1
  fnums[2] = 1
  fdens[2] = n
  count = 2
```

```
  пока fnums[count] <> 1 или fdens[count] <> 1
  нц
    count = count + 1
    fnums[count] = ((n + fdens[count - 2]) div fdens[count - 1]) * fnums[count -1] - fnums[count - 2]
    fdens[count] = ((n + fdens[count - 2]) div fdens[count - 1]) * fdens[count -1] - fdens[count - 2]
  кц
кон
```

**7.** Субфакториалом !n натурального числа n называется величина $!n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + .... + (-1)^n \frac{1}{n!})$. Разработайте алгоритм нахождения суммы субфакториалов чисел от *p* до *q*. (9-11 класс)

**Решение.** Первые два слагаемых в скобках равны, но имеют разный знак, поэтому их можно отбросить. Последнее слагаемое равно 1 для чётного *n* и -1 для нечётного *n*. Предпоследнее слагаемое по модулю равно *n*, а его знак противоположен знаку последнего слагаемого. Следующее слагаемое по модулю равно $(n - 1) \cdot n$, знак опять меняется на противоположный. Третье слагаемое равно $3 \cdot 4 \cdot ... \cdot (n - 1) \cdot n$.

Организуем цикл для всех *n* в диапазоне от *p* до *q*. Для каждого *n* вычисляем последнее слагаемое *s* – 1 или -1. Далее для всех *i* в диапазоне от *n* до 3 вычисляем очередное слагаемое по формуле $s \cdot i \cdot -1$ и прибавляем вновь вычисленное слагаемое к общей сумме.

```
алг СуммаСубфакториалов
нач
  цел p, q, n, sum, s, i

  ввод p, q

  если p > q или p < 1 или q < 1 то
    вывод "Некорректные исходные данные"
  иначе
    sum = 0
    для n от p до q
    нц
      если n mod 2 = 0 то
        s = 1
      иначе
        s = -1
      всё
      для i от n до 3 шаг -1
      нц
        s = s * i * -1
        sum = sum + s
      кц
    кц

    вывод sum
  всё
кон
```

**8.** Разработайте алгоритм, который определяет, является ли заданная матрица M ортонормированной, т. е. такой, в которой скалярное произведение каждой пары различных строк равно 0, а скалярное произведение каждой строки на себя равно 1. Матрица – прямоугольная таблица. (10-11 класс)

**Решение.** Необходимо разработать функцию, которая находит скалярное произведение двух массивов – перемножает элементы с одинаковыми индексами и суммирует эти произведения. Далее перебираем все строки матрицы, и, используя разработанную функцию, находим скалярное произведение строки на себя. Если для какой-то строки получаем результат, отличный от 1, завершаем работу программы с отрицательным результатом. Далее перебираем все возможные пары различных строк матрицы и считаем скалярное произведение каждой пары строк. Если для какой-то пары строк получаем результат, отличный от 0, завершаем работу программы с отрицательным результатом. Если же все проверки дали нужный результат, выводим сообщение, что матрица является ортонормированной.

```
алг ОртонормированнаяМатрица
нач
  вещ matr[m][n]
  цел m, n, i, j
  лог is_ortho

  ввод m, n

  если  m < 1 или n < 1 то
    вывод "Некорректные исходные данные"
  иначе
    для i от 1 до m
    нц
      для j от 1 до m
      нц
        ввод matr[i][j]
      кц
    кц

    is_ortho = истина
    i = 1
    пока i <= m и is_ortho
    нц
      если СкалярноеПроизведение(matr[i], matr[i]) <> 1 то        // В данном случае выражение matr[i]
        is_ortho = ложь                                           // представляет одномерный массив,
      всё                                                         // содержащий i-ую строку матрицы
      i = i + 1
    кц
    i = 1
    пока i <= m и is_ortho
    нц
      j = 1
      пока j <= n и is_ortho
      нц
        если СкалярноеПроизведение(matr[i], matr[j]) <> 0 то
          is_ortho = ложь
        всё
        j = j + 1
      кц
      i = i + 1
    кц

    если is_ortho то
      вывод "Матрица является ортонормированной"
    иначе
      вывод "Матрица не является ортонормированной"
    всё
  всё
кон

алг СкалярноеПроизведение(арг вещ x[n], y[n], арг цел n)
нач
  вещ p
  цел i

  p = 0
  для i от 1 до n
  нц
    p = p + x[i] * y[i]
  кц
  вернуть p
кон
```

**9.** Дан некоторый набор $A$ натуральных чисел: $a_1 < a_2 < \ldots < a_n$. Составьте алгоритм, который для любого не входящего в $A$ натурального числа $a$, $a_1 < a < a_n$, укажет ближайшее к нему снизу и ближайшее сверху числа из $A$. (9-11 класс)

**Решение.** Проверим правильность задания исходных данных – то, что число $a$ удовлетворяет условию $a_1 < a < a_n$. Далее, начиная с $i = 2$, находим первое число $a_i$ такое, что $a < a_i$. Поскольку набор чисел упорядочен и $a_1 < a$, можно утверждать, что найденное $a_i$ будет ближайшим к $a$ сверху, а число $a_{i-1}$ – ближайшим к $a$ снизу. Если при проверке обнаружится, что $a$ равно некоторому $a_i$, значит, исходные данные заданы неверно.

```
алг БлижайшееСнизуИСверху
нач
  цел A[n], n, i, a
  лог found

  ввод n
```

```
  для i от 1 до n
  нц
    ввод A[i]
  кц

  ввод a
  если A[1] >= a или a >= A[n] то
    вывод "Некорректные исходные данные"
  иначе
    found = ложь
    i = 2
    пока i <= n и не found
    нц
      если a < A[i] то
        вывод "Ближайшее снизу – ", A[i – 1], ", ближайшее сверху – ", A[i]
        found = истина
      всё
      если a = A[i] то
        вывод "Некорректные исходные данные"
        found = истина
      всё
    кц
  всё
кон
```

**10.** Вывести последовательность $d_k, d_{k-1}, \ldots, d_0$ десятичных цифр числа $3^{500}$, т.е. такую целочисленную последовательность, в которой каждый член $d_i$ удовлетворяет условию $0 \le d_i \le 9$ и $d_k \cdot 10^k + d_{k-1} \cdot 10^{k-1} + \ldots d_0 = 3^{500}$. (9-10 класс)

**Решение.** Число $3^{500}$ содержит более 200 цифр. Для представления такого числа необходимо разработать собственное представление данных и алгоритмы для реализации арифметических операций. Будем рассматривать только положительные числа. Цифры числа можно хранить в массиве, каждый элемент которого имеет размер один байт. Чтобы не хранить длину числа и не рассматривать разные случаи, будем заполнять все незанятые элементы массива незначащими нулями. Если результат какой-то операции не помещается в отведённое количество элементов массива, то он будет некорректен, но эта проблема существует во всех компьютерных системах.

Для сложения необходимо рассматривать цифры двух массивов, начиная с младшей, складывать каждую пару цифр и при необходимости осуществлять перенос в следующий разряд.

Для вычитания необходимо также рассматривать цифры двух массивов, начиная с младшей, вычитать одну цифру из другой и при необходимости осуществлять заём из следующего разряда.

Для умножения будем умножать первое число на каждую цифру второго числа, при этом результат умножения на вторую, третью и т.д. цифру нужно смещать вправо на 1, 2 и т.д. позиции. Результаты этих умножений надо сложить. Для умножения числа на одну цифру будем рассматривать цифры числа, начиная с младшей, и умножать каждую цифру на другую цифру, осуществляя перенос при необходимости.

Деление будем осуществлять, вычитая несколько раз делитель из делимого и считая количество вычитаний. Цикл прекращается, когда результат очередного вычитания станет меньше, чем делитель. Этот результат будет остатком от деления.

После реализации арифметических операций можно возводить в степень. Чтобы сократить число вычислений, можно использовать алгоритм быстрого возведения в степень, который основан на следующих свойствах степени: $x^y = (x^2)^{\frac{y}{2}}$ при чётном значении $y$ и $x^y = x \cdot x^{y-1}$ при нечётном значении $y$.

```
алг ВозведениеВСтепень(арг цел x, y)
нач
  цел p

  p = 1
  пока y <> 0
  нц
    если y mod 2 = 1 то
      p = p * x
      n = n – 1
    всё
    x = x * x
    n = n div 2
```

```
  кц
  вернуть р
кон
```