Материалы заданий Олимпиады школьников «Надежда энергетики» по предмету «информатика» в 2015/2016 учебном году

Характер и уровень сложности олимпиадных задач направлены на достижение целей проведения олимпиады: выявить способных участников, твердо владеющих школьной программой и наиболее подготовленных к освоению образовательных программ технических ВУЗов, обладающих логикой и творческим характером мышления, умеющих алгоритмически описать реальные ситуации из различных предметных областей и применить к ним наиболее подходящие методы информатики. Необходимы знания способов описания алгоритмов (язык блок-схем, псевдокод) и умение работать с базовыми конструкциями.

Задания Олимпиады дифференцированы по сложности и требуют различных временных затрат на полное и безупречное решение. Они охватывают все разделы школьной программы, но носят, в большинстве, комплексный характер, позволяющий варьировать оценки в зависимости от проявленных в решении творческих подходов и продемонстрированных технических навыков. Участники должны самостоятельно определить разделы и теоретические факты программы, применимые в каждой задаче, разбить задачу на подзадачи, грамотно выполнить решение каждой подзадачи, синтезировать решение всей задачи из решений отдельных подзадач.

Успешное выполнение олимпиадной работы не требует знаний, выходящих за пределы школьной программы, но, как видно из результатов Олимпиады, доступно не каждому школьнику, поскольку требует творческого подхода, логического мышления, умения увидеть и составить правильный и оптимальный план решения, четкого и технически грамотного выполнения каждой части решения.

Умение справляться с заданиями Олимпиады по информатике приходит к участникам с опытом, который вырабатывается на тренировочном и отборочном этапах Олимпиады.

Материалы заданий отборочного этапа Олимпиады школьников
«Надежда энергетики» по предмету «информатика»
в 2015/2016 учебном году

Задания для 9 классов

1. Заданы координаты N точек на плоскости (x_1, y_1), (x_2, y_2),…, (x_N, y_N). Определить координаты ромба со сторонами, параллельными осям координат, содержащего все эти точки. Вычислить площадь получившейся фигуры.

Решение. Поскольку стороны должны быть параллельны осям координат, можно найти крайние по оси *x* и по оси *y* точки, и через них провести прямые. Но в этом случае получится прямоугольник, который не является ромбом. Необходимо расширить его до квадрата с длиной стороны, равной длине большей стороны прямоугольника.

```
алг Ромб()
нач
  цел n, x[n], y[n]

  ввод n
  для i от 1 до n
  нц
    ввод x[i], y[i]
  кц

  minx = x[1]
  maxx = x[1]
  miny = y[1]
  maxy = y[1]
  для i от 2 до n
  нц
    если x[i] < minx то
      minx = x[i]
    всё
    если x[i] > maxx то
      maxx = x[i]
    всё
    если y[i] < miny то
      miny = y[i]
    всё
    если y[i] > maxy то
      maxy = y[i]
    всё
  кц

  если maxx – minx > maxy – miny то
    maxy = miny + maxx – minx
  всё
  если maxy – miny > maxx – minx то
    maxx = minx + maxy – miny
  всё

  вывод "Координаты ромба – (", minx, ", ", miny, "), (", maxx, ", ", miny, "), (",
                                 maxx, ", ", maxy, "), (", minx, ", ", maxy, ")"
  вывод "Площадь ромба – ", (maxx – minx) * (maxy – miny)
кон
```

2. Разработайте алгоритм для решения задачи: найти натуральные числа из диапазона от *N* до *M*, количество делителей у которых является произведением двух простых чисел.

Решение. Построим массив простых чисел с помощью решета Эратосфена. Далее для каждого числа в диапазоне от *N* до *M* найдём количество делителей и подбором проверим, не является ли количество делителей произведением двух простых чисел.

Для использования решета Эратосфена необходимо построить массив чисел в заданном диапазоне от 1 до *k*. 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 $\sqrt{k}$, 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*. Для удобства удалим нулевые элементы массива, чтобы найденные простые числа были расположены последовательно.

Для нахождения количества делителей будем проверять все числа *i* от 1 до $\sqrt{k}$ (*k* – число, для которого ищется количество делителей), и если *k* делится на очередное число, к количеству делителей можно прибавить 2, т.е. мы сразу находим два делителя – *i* и *k* / *i*.

```
алг Делители()
нач
  цел n, m, k, kol, i, j, p, q, nums[m]
  лог f

  ввод n, m

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

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

  k = 0
  для i от 2 до m
  нц
    если nums[i] <> 0 то
      k = k + 1
      nums[k] = nums[i]
    всё
  кц

  для p от n до m
  нц
    kol = 0
    для q от 1 до целая_часть(sqrt(p))
    нц
      если p mod q = 0 то
        kol = kol + 2
      всё
    кц
    f = ложь
    i = 1
    пока i <= k и не f
    нц
      j = i
      пока j <= k и не f
      нц
        если nums[i] * nums[j] = kol то
          f = истина
        всё
        j = j + 1
      кц
      i = i + 1
    кц
    если f то
      вывод p
    всё
  кц

кон
```

3. Функция Мёбиуса μ(n) определена для всех натуральных чисел n и принимает значения {-1, 0, 1} в зависимости от характера разложения числа n на простые сомножители:

- μ(n) = 1, если n свободно от квадратов (то есть не делится на квадрат никакого простого числа) и разложение n на простые сомножители состоит из чётного числа сомножителей; также μ(1) =1;
- μ(n) = -1, если n свободно от квадратов и разложение n на простые сомножители состоит из нечётного числа сомножителей;
- μ(n) = 0, если n не свободно от квадратов.

Разработайте алгоритм вычисления функции Мёбиуса μ(n).

Решение. Проверяем, что число свободно от квадратов. Считаем количество множителей. Далее по двум этим значениям выбираем значение функции Мёбиуса согласно приведённому выше условию.

Для проверки того, что число свободно от квадратов, и для поиска простых множителей необходимо наличие массива простых чисел. Для того чтобы использовать этот массив в двух функциях, будет удобнее объявить этот массив как глобальную переменную. Глобальные переменные – это такие переменные, которые обычно объявляются вне любой процедуры и функции, и при этом доступны во всей программе.

Существует ряд критериев качества программы. В первую очередь, это, конечно, корректность, надёжность и эффективность. Но не менее важным критерием является читабельность.

Любые средства, используемые в программе, должны улучшать качество программы. Использование глобальной переменной должно улучшать читабельность программы и упрощать её понимание. В данной задаче массив простых чисел является уникальной переменной, она действительно глобальна в программе, она требуется во всех частях программы, поэтому объявление такого массива как глобального является оправданным. Но ни в коем случае нельзя увлекаться глобальными переменными. Если объявлять все переменные как глобальные, то получится одна большая куча. Объявление глобальными переменных, которые нужны только в некоторой части программы, наоборот, ухудшит её читабельность и затруднит её понимание.

```
цел nums[10000], k                          // k – количество простых чисел

алг ФункцияМёбиуса(арг цел n)
нач
  цел n, k, m
  лог sf

  ввод n

  если n = 1 то
    m = 1
  иначе
    ПростыеЧисла(n)                         // Найдём простые числа в диапазоне от 1 до n
    sf = СвободноОтКвадратов(n)
    k  = КоличествоСомножителей(n)
    если не sf то
      m = 0
    иначе
      если k mod 2 = 0 то
        m = 1
      иначе
        m = -1
      всё
    всё
  всё
кон

алг ПростыеЧисла(арг цел n)
нач
```

```
  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

  k = 0
  для i от 2 до n
  нц
    если nums[i] <> 0 то
      k = k + 1
      nums[k] = nums[i]
    всё
  кц
кон

алг СвободноОтКвадратов(арг цел n)
нач
  цел i
  лог f

  f = истина
  i = 1
  пока nums[i] <= целая_часть(sqrt(n)) и f
  нц
    если n mod (nums[i] * nums[i]) = 0 то
      f = ложь
    всё
    i = i + 1
  кц

  вернуть f
кон

алг КоличествоСомножителей(арг цел n)
нач
  цел c, i

  c = 0
  i = 1
  пока n > 1
  нц
    если n mod nums[i] = 0 то
      c = c + 1
      n = n div nums[i]
    иначе
      i = i + 1
    всё
  кц

  вернуть c
кон
```

4. Полнократное число – положительное целое число, которое делится нацело квадратом каждого своего простого делителя. Разработайте алгоритм для нахождения полнократных чисел в диапазоне от *M* до *N*.

Решение. Можно заметить, что в список полнократных чисел будут входить квадраты всех чисел, третьи степени всех чисел, четвёртые степени всех чисел и т.д. Поэтому возьмём массив, заполненный нулями, и запишем в него сначала число 2 в степени 1, 2, 3…, потом число 3 в степени 1, 2, 3…, потом число 4 в степени 1, 2, 3… и т.д. (получить последовательно степени одного числа проще, чем одну и ту же степень разных чисел, т.к. в первом случае мы можем умножать текущий результат на нужное число, а во втором случае придётся каждый раз вычислять нужную степень через несколько умножений). Некоторые числа будут появляться повторно, но это не влияет на общий результат.

```
алг ПолнократныеЧисла()
нач
  цел m, n, i, j
  цел nums[n]

  ввод m, n

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

  nums[1] = 1
  для i от 2 до целая_часть(sqrt(n))
  нц
    v = i
    nums[v] = v
    пока v <= n
    нц
      v = v * i
      nums[v] = v
    кц
  кц

  для i от m до n
  нц
    если nums[i] <> 0 то
      вывод nums[i]
    всё
  кц
кон
```

5. Дано натуральное число n. Получить все такие натуральные q, чтобы n делилось нацело на q^2 и не делилось нацело на q^3.

Решение. Необходимо перебрать все возможные значения q и проверить делимость. Поскольку n должно быть больше или равно q^3 (иначе количество вариантов становится бесконечным), должно выполняться условие $q \le \sqrt[3]{n}$.

```
алг Делимость()
нач
  цел n, q

  ввод n

  для q от 2 до pow(n, 1 / 3)                  // pow(n, 1 / 3) – корень третьей степени из n
  нц
    если n mod (q * q) = 0 и n mod (q * q * q) <> 0 то
      вывод q
    всё
  кц
кон
```

Задания для 10 и 11 классов

1. В теории чисел четвёртая проблема Ландау звучит так: бесконечно ли множество простых чисел вида $n^2 + 1$, где n – натуральное число? Мы не просим Вас подтвердить или опровергнуть гипотезу. Вам предлагается разработать алгоритм для нахождения простых чисел указанного вида.

Решение. Ограничим диапазон поиска некоторым значением k. Построим массив простых чисел в диапазоне от 2 до k, используя решето Эратосфена. Затем для всех n от 1 до $\sqrt{k-1}$ проверим, что число $n^2 + 1$ является простым.

Для использования решета Эратосфена необходимо построить массив чисел в заданном диапазоне от 1 до k. 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 $\sqrt{k}$, 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 ПростыеЧислаЛандау()
нач
  цел k, i, j, nums[k]

  ввод k

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

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

  для n от 2 до целая_часть(sqrt(k – 1))
  нц
    если nums[n * n + 1] <> 0 то
      вывод n
    всё
  кц
кон
```

2. 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.

Решение. Можно найти разложение числа N 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 N в диапазоне от P до Q просмотреть все числа от 2 до $\sqrt{N}$ и проверить, делится ли N на квадрат какого-нибудь из них.

```
алг БесквадратныеЧисла()
нач
  цел p, q, n, k
  лог f

  ввод p, q

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

3. В теории чисел факториальным простым числом называется простое число, на единицу меньшее или на единицу большее факториала. Вам предлагается разработать алгоритм для нахождения простых чисел указанного вида.

Решение. Попробуем проверить числа на единицу большие или меньшие факториалов всех чисел в диапазоне от 1 до *n*. Значения факториалов расположены достаточно далеко друг от друга и строить массив простых чисел нет смысла, проще проверить на простоту только нужные значения.

```
алг ФакториальныеПростыеЧисла()
нач
  цел n, i, f

  ввод n

  для i от 1 до n
  нц
    f = factorial(i)
    если Простое(f - 1) то
      вывод f - 1
    всё
    если Простое(f + 1) то
      вывод f + 1
    всё
  кц
кон

алг Простое(арг цел N)
нач
  цел i

  если N <= 1 то
    вернуть ложь
  всё
  если N = 2 или  N = 3 или  N = 5 или  N = 7 то
    вернуть истина
  всё
  если N mod 2 = 0 то
    вернуть ложь
  всё
  если N mod 3 = 0 то
    вернуть ложь
  всё

  для i от 5 до целая_часть(sqrt(N)) шаг 6               // Рассматриваем числа, меньшие корня (!) из N
  нц
    если N mod i = 0 то
```

```
      вернуть ложь
    всё
    если N mod (i + 2) = 0 то
      вернуть ложь
    всё
  кц
  вернуть истина
кон

алг factorial(арг цел n)
нач
  цел f

  f = 1
  для i от 1 до n
  нц
    f = f * i
  кц
  вернуть f
кон
```

4. В теории чисел бинарная проблема Гольдбаха звучит так: любое чётное число, начиная с 4, можно представить в виде суммы двух простых чисел. Мы не просим Вас подтвердить или опровергнуть гипотезу. Вам предлагается разработать алгоритм для представления чётных чисел в диапазоне от m до n указанным способом.

Решение. Построим массив простых чисел в диапазоне от 1 до *n*. Далее для каждого числа от *m* до *n* будем перебирать все пары простых чисел из построенного массива, и проверять, не равна ли сумма этой пары текущему числу.

Для использования решета Эратосфена необходимо построить массив чисел в заданном диапазоне от 1 до *k*. 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 $\sqrt{k}$, 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*. Для удобства удалим нулевые элементы массива, чтобы найденные простые числа были расположены последовательно.

```
алг БинарнаяПроблемаГольдбаха()
нач
  цел m, n, k, i, j, p, nums[n]

  ввод m, n

  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

  k = 0
  для i от 2 до n
```

```
  нц
    если nums[i] <> 0 то
      k = k + 1
      nums[k] = nums[i]
    всё
  кц

  если m < 4 то
    m = 4
  всё
  если m mod 2 = 1 то
    m = m + 1
  всё

  для p от m до n шаг 2
  нц
    для i от 1 до k
    нц
      для j от i до k
      нц
        если nums[i] + nums[j] = p то
          вывод p, " = ", nums[i], " + ", nums[j]
        всё
      кц
    кц
  кц
кон
```

5. В теории чисел число Вудала (*Wn*) – любое натуральное число вида $n \cdot 2^n - 1$ для некоторого натурального *n*. Числа Вудала, являющиеся простыми числами, называются простыми числами Вудала. Вам предлагается разработать алгоритм для нахождения простых чисел Вудала в диапазоне от *P* до *Q*.

Решение. Для того чтобы уменьшить перебор будем отталкиваться от числа *n*. Сначала подберём наименьшее значение *n* такое, чтобы число W_n было больше или равно *P*. Затем будет увеличивать *n* на 1, пока число W_n будет меньше или равно *Q*. Для каждого вычисленного числа W_n будем проверять, является ли оно простым.

```
алг ПростыеЧислаВудала()
нач
  цел p, q, n, w

  ввод p, q

  n = 0
  выполнить
    n = n + 1
    w = n * 2 ^ n – 1
  до w >= p

  пока w <= q
  нц
    если Простое(w) то
      вывод w
    всё
    n = n + 1
    w = n * 2 ^ n – 1
  кц

кон

алг Простое(арг цел N)
нач
  цел i

  если N <= 1 то
    вернуть ложь
  всё
```

```
  если N = 2 или  N = 3 или  N = 5 или  N = 7 то
    вернуть истина
  всё
  если N mod 2 = 0 то
    вернуть ложь
  всё
  если N mod 3 = 0 то
    вернуть ложь
  всё

  для i от 5 до целая_часть(sqrt(N)) шаг 6               // Рассматриваем числа, меньшие корня (!) из N
  нц
    если N mod i = 0 то
      вернуть ложь
    всё
    если N mod (i + 2) = 0 то
      вернуть ложь
    всё
  кц
  вернуть истина
кон
```