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
```