ЗАДАНИЕ ПО ИНФОРМАТИКЕ
ВАРИАНТ 73101 для 10 класса

Для заданий 1, 2, 4, 5 требуется разработать алгоритмы на языке блок-схем, псевдокоде или естественном языке

1. Для проверки, является ли большое целое простым, может использоваться вероятностный тест Ферма. Пусть $p > 2$ – проверяемое число. Тогда:
   - случайно выбираем a: $2 \le a \le p - 2$;
   - если $a^{p-1} \neq 1$ $\{\text{mod } p\}$, то p – составное.

   В тесте Ферма эти проверки выполняются для t случайно выбираемых a.

   Написать алгоритм проверки вводимого числа на простоту по тесту Ферма.

   Примечание: $x = y$ $\{\text{mod } n\}$, если существует целое k, для которого $x = y + k \bullet n$.

Решение. В цикле t раз выбираем число a в заданном диапазоне. Для каждого a проверяем условие $a^{p-1} \neq 1$ $\{\text{mod } p\}$. Для того чтобы при возведении в степень не получилось слишком большого числа (выходящего за пределы чисел, представимых в компьютере), можно применять операцию вычисления остатка от деления после каждого умножения на a. Если для какого-то значения a результат не равен 1, то число – составное, и проверку можно прекращать.

Выполняется следующее равенство: $(x \cdot y) \text{ mod } p = ((x \text{ mod } p) \cdot (y \text{ mod } p)) \text{ mod } p$

Поскольку в нашем случае $a < p$, то $a \text{ mod } p = a$. Значит, на первом шаге цикла мы имеем результат операции $a^n \text{ mod } p$ (для $n = 1$). Тогда $a^{n+1} \text{ mod } p = ((a^n \text{ mod } p) \cdot (a \text{ mod } p)) \text{ mod } p$. $a^n \text{ mod } p$ мы уже имеем, и $a \text{ mod } p = a$. Остаётся только выполнить умножение и взятие остатка от деления.

```
алг ТестФерма()
нач
  цел p, a, t, i, r
  лог prime

  ввод p, t
  если p <= 2 или t <= 0 то
    вывод "Некорректные исходные данные"
  иначе

    prime = истина
    i = 1
    пока i <= t и prime
    нц
      генерация a в диапазоне от 2 до p - 2
      r = a
      для j от 1 до p – 2
      нц
        r = (r * a) mod p
      кц
      если r <> 1 то
        prime = ложь
      всё
    кц

    если prime то
      вывод "Число простое"
    иначе
      вывод "Число составное"
    всё
  всё

кон
```

2. В пансионате для спортивного досуга детей оборудована специальная площадка с большим числом крупных клеток L, выложенных в дорожки одинаковой длины. По дорожкам (от начала до конца) с клетки на клетку любят прыгать отдыхающие дети. На каждой клетке нарисован вес – натуральное число. Выигрывает тот ребенок, который при прыжках набрал минимальный суммарный вес. В игре принимали участие $M \leq L$ детей, прыгающих за один ход на 1, 3, 4 или 5 клеток. Предложите наиболее оптимальный способ обработки и хранения информации для моделирования ситуации (например, для определения победителя). Примечание: прыгать на 2 клетки нельзя.

Решение. Для хранения данных будем использовать матрицу из $L \times L$ элементов, в элементах которой записан вес каждой клетки. Результаты детей запишем в виде массива из $L \times M$ элементов, причём элемент равен 1, если ребёнок прыгал на данную клетку, и 0 – в противном случае. Также надо записать номер дорожки, по которой он прыгал. Чтобы получить итоговый результат, надо сложить веса клеток, на которые прыгал каждый ребёнок. Для проверки соблюдения правил будем вычислять разницу между индексами элементов в каждой строке второй матрицы, содержащих значение 1. Для этого положим сначала номер предыдущей использованной клетки 0, затем при нахождении в массиве значения 1 вычисляем разницу. Если она не равна 1, 3, 4 или 5, значит, правила была нарушены. Иначе прибавляем к общему результату вес текущей клетки и запоминаем номер текущей клетки как номер предыдущей.

```
алг СпортивныйДосуг()
нач
  цел l, m, weights[l, l], tracks[m], jumps[l, m], results[m], i, j, prev
  лог right

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

    для i от 1 до l
    нц
      для j от 1 до l
      нц
        ввод weights[i, j]
      кц
    кц

    для i от 1 до m
    нц
      ввод tracks[i]
    кц

    для i от 1 до l
    нц
      для j от 1 до m
      нц
        ввод jumps[i, j]
      кц
    кц

    для i от 1 до m
    нц
      results[i] = 0
      right = истина
      prev = 0
      j = 1
      пока j <= l и right
      нц
        если weights[tracks[i], j] = 1 то
          если j – prev = 1 или j – prev = 3 или j – prev = 4 то
            results[i] = results[i] + weights[tracks[i], j]
            prev = j
          иначе
            right = ложь
          всё
```

```
        всё
      кц

      если right то
        results[i] = -1
      всё
    кц

    для i от 1 до m
    нц
      если results[i] = -1 то
        вывод "Ребёнок нарушил правила"
      иначе
        вывод results[i]
      всё
    кц

  всё
кон
```

3. Десятиклассник Сережа любит играть с калькулятором. Он часто сначала делит вещественные числа a и b друг на друга, а затем результат умножает на b. Выполнив эти действия много раз (сначала много делений, а затем столько же умножений), Сережа получил в результате некоторое число. Будет ли оно исходным? Объясните, почему?

Решение. Не будет. Возникающая в процессе вычислений погрешность, обусловленная округлением вещественных чисел из-за ограниченного количества знаков в числе при представлении в ЭВМ, изменит результат.

4. Не используя дополнительный массив или простые методы сортировок, найти в матрице номера двух первых минимальных элементов.

Решение. Найдём минимальный элемент матрицы. Переставим его на первое место в матрице и снова найдём минимальный элемент матрицы из элементов, кроме первого.

Данный алгоритм рассчитан на поиск именно двух минимальных элементов. В аналогичной задаче другого варианта приведён другой алгоритм, который может быть использован для поиска любого количества минимумов.

```
алг ДваМинимума()
нач
  цел m, n, x[m, n], imin[2], jmin[2], k, i, j, y

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

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

    // Находим первый минимум
    imin[1] = 1
    jmin[1] = 1
    для i от 1 до m
    нц
      для j от 1 до n
      нц
        если x[i, j] < x[imin[1], jmin[1]] то
          imin[1] = i
          jmin[1] = j
        всё
      кц
    кц
```

```
    k = 1

    // Меняем местами найденный минимум и элемент x[1, 1]
    y = x[1, 1]
    x[1, 1] = x[imin[1], jmin[1]]
    x[imin[1], jmin[1]] = y
    // В качестве начального приближения берём элемент x[1, 2]
    imin[2] = 1
    jmin[2] = 2
    // Просматриваем первую строку матрицы, начиная с 3-го элемента
    для j от 3 до n
    нц
      если x[1, j] < x[imin[2], jmin[2]] то
        imin[1] = 1
        jmin[1] = j
      всё
    кц
    // Просматриваем остальную часть матрицы
    для i от 2 до m
    нц
      для j от 1 до n
      нц
        если x[i, j] < x[imin[2], jmin[2]] то
          imin[2] = i
          jmin[2] = j
        всё
      кц
    кц

    вывод "Индексы первого минимального элемента - ", imin[1], ", ", jmin[1]
    вывод "Индексы второго минимального элемента - ", imin[2], ", ", jmin[2]

  всё
кон
```

5. Число Фибоначчи – натуральное число, удовлетворяющее следующим соотношениям: $F_0 = 1$, $F_1 = 1$, $F_n = F_{n\text{-}1} + F_{n\text{-}2}$, $n \ge 2$. Даны целые числа n и m ($1 \le n \le 10^{18}$, $2 \le m \le 10^5$), необходимо найти остаток от деления n-го числа Фибоначчи на m.

Решение. Можно найти n-ое число Фибоначчи и затем остаток от его деления на m. Если же мы опасаемся, что n-ое число Фибоначчи окажется слишком большим для представления в компьютере, можно воспользоваться свойствами операции «остаток от деления» и написать рекурсивную функцию $[F_n]_m = [[F_{n-1}]_m + [F_{n-2}]_m]_m$. Впрочем, рекурсия тоже требует много затрат, поэтому лучше заменить её на итерацию.

```
алг ОстаткиОтДеленияЧиселФибоначчи()
нач
  цел m, n, f0, f1, f, k

  ввод n, m
  если n <= 0 или n > 10e18 или m <= 1 или m > 10e5 то
    вывод "Некорректные исходные данные"
  иначе

    // Остаток от деления 1 на любое m ≥ 2 равен 1.
    // Таким образом, переменные f1 и f содержат не числа Фибоначчи, а остатки от деления.
    f1 = 1
    f  = 1
    k = 1                          // Номер найденного остатка от деления
    пока k < n
    нц
      f0 = f1
      f1 = f
      f  = (f0 + f1) mod m
      k = k + 1
    кц

    вывод "Остаток от деления ", n, "-го числа Фибоначчи на ", m, " равен ", f

  всё
кон
```

ЗАДАНИЕ ПО ИНФОРМАТИКЕ
ВАРИАНТ 73102 для 10 класса

Для заданий 1, 2, 4, 5 требуется разработать алгоритмы на языке блок-схем, псевдокоде или естественном языке

1. Для проверки, является ли большое целое простым, может использоваться вероятностный тест Ферма. Пусть $p > 2$ – проверяемое число. Тогда:
   - случайно выбираем a: $2 \le a \le p - 2$;
   - если $a^{p-1} \ne 1$ $\{\mathrm{mod}\ p\}$, то p – составное.

   В тесте Ферма эти проверки выполняются для t случайно выбираемых a.

   Написать алгоритм проверки вводимого числа на простоту по тесту Ферма.

   Примечание: $x = y$ $\{\mathrm{mod}\ n\}$, если существует целое k, для которого $x = y + k \bullet n$.

Решение. В цикле t раз выбираем число a в заданном диапазоне. Для каждого a проверяем условие $a^{p-1} \ne 1$ $\{\mathrm{mod}\ p\}$. Для того чтобы при возведении в степень не получилось слишком большого числа (выходящего за пределы чисел, представимых в компьютере), можно применять операцию вычисления остатка от деления после каждого умножения на a. Если для какого-то значения a результат не равен 1, то число – составное, и проверку можно прекращать.

Выполняется следующее равенство: $(x \cdot y) \bmod p = ((x \bmod p) \cdot (y \bmod p)) \bmod p$

Поскольку в нашем случае $a < p$, то $a \bmod p = a$. Значит, на первом шаге цикла мы имеем результат операции $a^n \bmod p$ (для $n = 1$). Тогда $a^{n+1} \bmod p = ((a^n \bmod p) \cdot (a \bmod p)) \bmod p$. $a^n \bmod p$ мы уже имеем, и $a \bmod p = a$. Остаётся только выполнить умножение и взятие остатка от деления.

```
алг ТестФерма()
нач
  цел p, a, t, i, r
  лог prime

  ввод p, t
  если p <= 2 или t <= 0 то
    вывод "Некорректные исходные данные"
  иначе

    prime = истина
    i = 1
    пока i <= t и prime
    нц
      генерация a в диапазоне от 2 до p - 2
      r = a
      для j от 1 до p – 2
      нц
        r = (r * a) mod p
      кц
      если r <> 1 то
        prime = ложь
      всё
    кц

    если prime то
      вывод "Число простое"
    иначе
      вывод "Число составное"
    всё
  всё

кон
```

2. В пансионате для спортивного досуга детей оборудована специальная площадка с большим числом крупных клеток L, выложенных в дорожки одинаковой длины. По дорожкам (от начала до конца) с клетки на клетку любят прыгать отдыхающие дети. На каждой клетке нарисован вес – натуральное число. Выигрывает тот ребенок, который при прыжках набрал минимальный суммарный вес. В игре принимали участие $M \leq L$ детей, прыгающих за один ход на 1, 2, 4 или 5 клеток. Предложите наиболее оптимальный способ обработки и хранения информации для моделирования ситуации (например, для определения победителя). Примечание: прыгать на 3 клетки нельзя.

Решение. Для хранения данных будем использовать матрицу из $L \times L$ элементов, в элементах которой записан вес каждой клетки. Результаты детей запишем в виде массива из $L \times M$ элементов, причём элемент равен 1, если ребёнок прыгал на данную клетку, и 0 – в противном случае. Также надо записать номер дорожки, по которой он прыгал. Чтобы получить итоговый результат, надо сложить веса клеток, на которые прыгал каждый ребёнок. Для проверки соблюдения правил будем вычислять разницу между индексами элементов в каждой строке второй матрицы, содержащих значение 1. Для этого положим сначала номер предыдущей использованной клетки 0, затем при нахождении в массиве значения 1 вычисляем разницу. Если она не равна 1, 2, 4 или 5, значит, правила была нарушены. Иначе прибавляем к общему результату вес текущей клетки и запоминаем номер текущей клетки как номер предыдущей.

```
алг СпортивныйДосуг()
нач
  цел l, m, weights[l, l], tracks[m], jumps[l, m], results[m], i, j, prev
  лог right

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

    для i от 1 до l
    нц
      для j от 1 до l
      нц
        ввод weights[i, j]
      кц
    кц

    для i от 1 до m
    нц
      ввод tracks[i]
    кц

    для i от 1 до l
    нц
      для j от 1 до m
      нц
        ввод jumps[i, j]
      кц
    кц

    для i от 1 до m
    нц
      results[i] = 0
      right = истина
      prev = 0
      j = 1
      пока j <= l и right
      нц
        если weights[tracks[i], j] = 1 то
          если j – prev = 1 или j – prev = 2 или j – prev = 4 то
            results[i] = results[i] + weights[tracks[i], j]
            prev = j
          иначе
            right = ложь
          всё
```

```
        всё
      кц

      если right то
        results[i] = -1
      всё
    кц

    для i от 1 до m
    нц
      если results[i] = -1 то
        вывод "Ребёнок нарушил правила"
      иначе
        вывод results[i]
      всё
    кц

  всё
кон
```

3. Десятиклассник Сережа любит играть с калькулятором. Он часто сначала делит вещественные числа a и b друг на друга, а затем результат умножает на b. Выполнив эти действия много раз (сначала много делений, а затем столько же умножений), Сережа получил в результате некоторое число. Будет ли оно исходным? Объясните, почему?

Решение. Не будет. Возникающая в процессе вычислений погрешность, обусловленная округлением вещественных чисел из-за ограниченного количества знаков в числе при представлении в ЭВМ, изменит результат.

4. Не используя дополнительный массив или простые методы сортировок, найти в матрице значения трех первых минимальных элементов.

Решение. Найдём минимальный элемент матрицы. Проверим, сколько таких элементов существует в матрице. Если их 3 и больше, значит, задача решена. Если нет, ещё раз найдём минимальный элемент из тех элементов, которые больше первого минимума. В случае необходимости найдём также третий минимальный элемент из тех элементов, которые больше первых двух минимумов.

Приведённый алгоритм может быть использован для поиска любого количества минимумов.

```
алг ТриМинимума()
нач
  цел m, n, x[m, n], min[m * n], k, km, max, i, j

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

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

    // Находим минимальное значение в матрице, количество таких значений и максимальное значение в матрице
    min[1] = x[1, 1]
    km = 1
    max = x[1, 1]
    для i от 1 до m
    нц
      для j от 1 до n
      нц
        если x[i, j] < min[1] то
          min[1] = x[i, j]
```

```
        km = 1
      иначе если x[i, j] = min[1] то
        km = km + 1
      иначе если x[i, j] > max то
        max = x[i, j]
      всё
      всё
      всё
    кц
  кц

  // Дублируем минимальное значение столько раз, сколько оно встречается в матрице
  для i от 2 до km
  нц
    min[i] = min[1]
  кц
  k = km

  пока k < 3
  нц
    // Ищем минимальное значение среди значений, которые больше последнего найденного минимума
    // Поскольку последний найденный минимум не меньше предыдущих, элемент матрицы, который больше
    // последнего минимума, будет больше всех найденных минимумов.
    // Чтобы не искать первое значение, которое больше последнего найденного минимума, используем
    // для инициализации максимальное значение в матрице, что не повлияет на результат.
    min[k + 1] = max
    km = 1
    для i от 1 до m
    нц
      для j от 1 до n
      нц
        если x[i, j] > min[k] и x[i, j] < min[k + 1] то
          min[k + 1] = x[i, j]
          km = 1
        иначе если x[i, j] = min[k + 1] то
          km = km + 1
        всё
        всё
      кц
    кц
    если min[k + 1] = max то
      km = km – 1
    всё

    // Дублируем минимальное значение
    для i от 2 до km
    нц
      min[k + i] = min[k + 1]
    кц
    k = k + km
  кц

  для i от 1 до 3
  нц
    вывод i, "-й минимум равен ", min[i]
  кц

всё
кон
```

5. Число трибоначчи – натуральное число, удовлетворяющее следующим соотношениям: $t_0 = 1$, $t_1 = 0$, $t_2 = 1$, $t_{n+3} = t_{n+2} + t_{n+1} + t_n$, $n \geq 2$. Даны целые числа n и m ($1 \leq n \leq 10^{18}$, $2 \leq m \leq 10^5$), необходимо найти остаток от деления n-го числа трибоначчи на m.

Решение. Можно найти n-ое число трибоначчи и затем остаток от его деления на m. Если же мы опасаемся, что n-ое число трибоначчи окажется слишком большим для представления в компьютере, можно воспользоваться свойствами операции «остаток от деления» и написать рекурсивную функцию $[t_{n+3}]_m = [[t_{n+2}]_m + [t_{n+1}]_m + [t_n]_m]_m$. Впрочем, рекурсия тоже требует много затрат, поэтому лучше заменить её на итерацию.

```
алг ОстаткиОтДеленияЧиселТрибоначчи()
нач
  цел m, n, t0, t1, t2, t, k
```

```
  ввод n, m
  если n <= 0 или n > 10e18 или m <= 1 или m > 10e5 то
    вывод "Некорректные исходные данные"
  иначе

    // Остаток от деления 0 на любое m ≥ 2 равен 0, а остаток от деления 1 на любое m ≥ 2 равен 1.
    // Таким образом, переменные t1, t2 и t содержат не числа трибоначчи, а остатки от деления.
    t1 = 0
    t2 = 0
    t  = 1
    k = 2                               // Номер найденного остатка от деления
    пока k < n
    нц
      t0 = t1
      t1 = t2
      t2 = t
      t  = (t0 + t1 + t2) mod m
      k = k + 1
    кц

    если n = 1 то
      вывод "Остаток от деления 1-го числа Фибоначчи на ", m, " равен 0"
    иначе
      вывод "Остаток от деления ", n, "-го числа трибоначчи на ", m, " равен ", t
    всё

  всё
кон
```