Разбор заданий муниципального этапа
Всероссийской олимпиады школьников по информатике
2019-2020 учебный год
9-11 классы

Задача A. Кино

Время, затраченное Мишей, складывается из времени ожидания начала сеанса и времени просмотра фильма.

Если Миша пришел на кинофестиваль в M минут, то время ожидания каждого фильма будет следующим:

- фильм A: $(15 - M \bmod 15)$
  но при этом, если Миша пришел ровно в минуту начала фильма, то данная формула вычислит время ожидания 15, а должно быть 0. Чтобы это исправить, возьмем от этой величины остаток от деления на 15 (все числа, меньше 15, останутся неизменными, а число 15 после взятия остатка станет равным нулю).
  В итоге время ожидания фильма A: $(15 - M \bmod 15) \bmod 15$
- аналогично время ожидания фильма B: $(10 - M \bmod 10) \bmod 10$
- время ожидания фильма C: $(5 - M \bmod 5) \bmod 5$

Сложим найденные значения с длительностью соответствующего фильма, а затем найдем наименьшее из них.

Пример программы на языке Pascal

```
program task_A;
var x, y, z, M: integer;
begin
read (x);
read (y);
read (z);
read (M);
write( min (x + (15 - M mod 15) mod 15,
       min( y + (10 - M mod 10) mod 10,
            z + (5 - M mod 5) mod 5)));
end.
```

Задача B. Пароль

Формализуя условие задачи, понимаем, что требуется по числу N восстановить наименьшее число, произведение цифр которого равно N.

Чтобы число было наименьшим, следует стремиться выполнить два условия (причем именно в таком приоритете):

1) количество цифр в числе должно быть наименьшим
2) меньшие цифры должны стоять в старших разрядах.

Для получения цифр искомого числа, будем раскладывать N на множители-цифры. Чтобы цифр получилось как можно меньше, будем начинать разложение с наибольшей цифры, т.е. с 9. Так как сразу выводить на экран большие множители мы не можем (большие цифры нужно будет поставить в конец числа), то организуем подсчет количества цифр каждой величины. Для этого заведем массив с индексами от 2 до 9 (цифры 0 не может быть в числе, так как тогда произведение будет равно нулю, а у нас $N \geq 1$; цифра 1 нам в числе не нужна, так как она не влияет на произведение, но при этом добавляет лишний разряд в число, тем самым увеличивая его). Каждый элемент массива будет хранить количество множителей в числе, равных индексу элемента. После того, как количество сомножителей каждого вида будет посчитано, выводим их на экран нужное количество раз, начиная с меньшей цифры (т.е. фактически реализуем сортировку подсчетом).

Осталось обработать только два крайних случая:

1) если $N < 10$, то наш алгоритм выдаст в качестве ответа само число N, а это неверно, так как по условию пароль должен содержать хотя бы две цифры. Очевидно, что при таком условии наименьшим будет двузначное число, у которого в старшем разряде единица, а в младшем – N, т.е. число $10 + N$
2) некоторые числа не могут быть получены как произведение только цифр (например, число $22 = 11 \cdot 2$, но 11 не является цифрой и не может быть разложено на другие множители); значит, для таких чисел программа должна выдавать ответ «`No solution`»

Пример программы на языке Pascal

```
program task_B;
var n, i, j, k: longint;
    del: array [2..9] of integer;
begin
for  i := 2 to 9 do
   del[i] := 0;
read (n);
if (n < 9) then
   write (10 + n)
else
  begin
```

```
 i := 9;
 while (n > 9) and (i >= 2) do
   if (n mod i = 0) then
     begin
     del [i] := del[i] + 1;
     n := n div  i;
     end
  else
      i := i - 1;
  if (n > 9) then
    write ('No solution')
  else
     begin
     del[n] := del [n] + 1;
     for j := 2 to 9 do
       for k := 1 to del[j] do
           write (j);
     end;
  end;
end.
```

Задача C. Ожерелье и браслеты

Для решения требуется искать подряд идущие золотые бусины (подряд идущие единицы во входной последовательности) и вычислять, сколько последовательностей длины k может из них получиться, используя операцию целочисленного деления.

Также следует учитывать, что ожерелье представляет собой замкнутую окружность, а при вводе она «развернута» в последовательность. Т.е. бусины, расположенные в начале и в конце введенной последовательности, на самом деле замкнуты в одну последовательность, из которой тоже может получиться несколько браслетов. Например, для последовательности

1 1 0 0 1 1 1 1 0 1

могут получиться два браслета из трех бусин: один браслет - из бусин в центре (выделены красным), и один - из бусин по краям после их замыкания в окружность (выделены синим).

Пример программы на языке Pascal

```
program task_D;
var  i, j, n, k, a, gold, e , friend: longint;
      integer;
     f:boolean;
begin
read(n, k);
gold := 0; friend := 0;
j := 0; f := true;
for i := 1 to n do
  begin
  read(a);
  if a = 0 then f := false;
  if f then j := j + 1;
  if (not f ) then
     if a = 1 then
        gold := gold + 1
     else
        begin
        friend := friend + gold div k;
        gold := 0;
        end;
  end;
if a = 1 then
   friend := friend + (j + gold) div k
else
  friend := friend + j  div k;
writeln( friend);
end.
```

Задача D. Магические слова

Разделим данное нам слово на блоки, состоящие из максимального количества подряд идущих букв одного типа (типом буквы назовём её гласность/согласность). Очевидно, что ответом для всего слова будет сумма ответов для каждого блока.

Разберем решение для фиксированного блока. Пусть есть k подряд идущих букв одного типа. Нужно разделить их на минимальное количество блоков длины меньше трёх. Значит, максимально могут идти подряд две однотипных буквы, т.е. нужно посчитать количество пар в этом блоке. Очевидно, что если k – четное, то количество пар вычисляется как $k\ div\ 2$, а если нечетное, то один символ останется без пары, но его всё равно требуется отделять. Поэтому количество «пар» в этом случае будет $k\ div\ 2 + 1$. Для обоих случаев это количество можно вычислить по формуле $(k + 1)\ div\ 2$. Тогда количество символов, необходимых для разбиения на такие блоки, на один меньше, то есть $(k + 1)\ div\ 2 - 1$.

Для разбиения на блоки подряд идущих гласных / согласных букв можно проверять каждый символ слова (для этого можно записать все гласные и согласные буквы в отдельную структуру – строку, множество и т.д. или проверять их коды) и вычислять длины последовательностей из подряд идущих согласных и гласных.

Пример программы на языке Pascal

```
program task_D;
var  i, n, c, v : integer;
     s, gl, sogl: string;
begin
read(s);
sogl := 'bcdfghjklmnpqrstvwxz';
gl := 'aeiouy';
n := 0;
c := 0;
v := 0;
for i := 1 to length (s) do
  begin
  if pos (s [i], gl) <> 0 then
    begin
    c := c + 1;
    if (i < length (s)) and (pos(s [i + 1], sogl) <> 0)
                or (i = length (s)) then
      begin
      n := n + (c + 1) div 2 - 1;
      c := 0;
      end;
    end;
  if pos (s [i], sogl) <> 0 then
    begin
```

```
    v := v + 1;
    if (i < length(s)) and (pos(s [i + 1], gl) <> 0)
            or (i = length(s)) then
      begin
      n := n + (v + 1) div 2 - 1;
      v := 0;
      end;
    end;

  end;
 writeln(n);
end.
```

Задача E. Расселение в отеле

Для решения задачи (с учетом величины исходных данных) достаточно реализовать описанный поиск.

Для выбора подходящего номера найдем один номер, находящийся левее ресепшн, и один номер, находящийся правее ресепшн. Если окажется, что найден номер только с одной стороны, то он и будет искомым.

Если найдены номера с обеих сторон, продолжаем выбор: если вместимость обоих равна нужному значению (или вместимость обоих не равна нужному значению), то выбираем номер, находящийся ближе в ресепшн. Если при этом и расстояние одинаковое, то выбираем номер слева.

После того, как номер будет найден, нужно отметить, что он занят (и не может быть использован для следующих групп). Это можно сделать, например, обнулив вместимость номера.

Пример программы на языке Pascal

```
program tasm_E;
var n, k, m, i, j, l, r, ans, b: integer;
  a : array [1..1000] of integer;
begin
read(n, k, m);
for i := 1 to n do
  read(a[i]);
for j := 1 to k do
  begin
  read(b);
  l := -1;
  r := -1;
  //подходящий номер левее
  for i := m downto 1 do
    begin
    if (a[i] = b) and ((l = -1) or (l <> -1)
                                    and (a[l] <> b)) then
      l := i;
    if (a[i] > b) and (l = -1) then
      l := i;
    end;
  //подходящий номер правее
  for i := m + 1 to n do
    begin
    if (a[i] = b) and ((r = -1) or (r <> -1)
                                     and (a[r] <> b)) then
      r := i;
    if (a[i] > b) and (r = -1) then
```

```
      r := i;
    end;
    ans := -1;
    if (l = -1) and (r = -1) then
       ans := -1
    else
       begin
       //если нашлись номера с обеих сторон от входа
       if (l <> - 1) and (r <> -1) then
             begin
             //если номера слева и справа идентичны,
             //ищем ближайший
             if  ((a[l] = b) and (a[r] = b) or (a[l] <> b)
                    and (a[r] <> b)) and (m - l >= r - m) then
                 ans := r
             else
                 ans := l;
             //номер слева ровно на b гостей, а справа нет
             if (a[l] = b) and (a[r] > b) then ans := l;
             //номер справа ровно на b гостей, а слева нет
             if (a[l] > b) and (a[r] = b) then ans := r;
             end
        else
         //если нашли номер только с одной стороны
           begin
           if l <> -1 then ans := l;
           if r <> -1 then ans := r;
           end;
     if (ans <> - 1) then
         a[ans] := 0; //номер занят
     end;
  write(ans, ' ');
  end;
end.
```