РАЙОННАЯ ОЛИМПИАДА ПО ПРОГРАММИРОВАНИЮ

8 декабря 2016 года

Задача 1. «Речные гонки»

- Егор и Петр участвуют в речной гонке на лодках: участники одновременно стартуют в пункте А и должны проплыть против течения реки в пункт В. Тот, кто приплывет первым, объявляется победителем. В результате жеребьевки Егору и Петру выпало участвовать в одном заплыве.
- Нам известны скорости движения лодок ребят и скорость течения реки.
- Ваша задача – определить, кто же из них победит.

Задача 1. «Речные гонки»

- Входной файл **river.in**:

В первой строке ввода содержатся три целых числа: **V1** – скорость лодки Егора, **V2** – скорость лодки Петра и **V3** – скорость течения реки ($0 \le V1, V2, V3 \le 10^6$).

- Выходной файл **river.out**:

Выводится одна строка, содержащая следующий текст:

- "EGOR", если первым приплывет Егор;
- "PETR", если первым приплывет Петр;
- "RIVER", если победителя определить не удалось.

Задача 1. «Речные гонки»

Примеры:

river.in	river.out
10 5 1	EGOR
4 5 2	PETR
6 6 2	RIVER

Задача 1. «Речные гонки»

- Победитель будет выявлен в том случае, если скорость одного из мальчиков будет больше скорости другого и больше скорости реки.
- Если же скорости мальчиков одинаковы или скорости обоих мальчиков меньше скорости реки, то победителя нет.

Задача 1. «Речные гонки»

```
var
  v1, v2, v3 : longint;
  f_in, f_out : text;
begin
  assign(f_in,'river10.in');  reset(f_in);
  readln(f_in,v1,v2,v3);  close(f_in);
  assign(f_out,'river.out');  rewrite(f_out);
  if (v1>v2) and (v1>v3) then write(f_out,'EGOR');
  if (v2>v1) and (v2>v3) then write(f_out,'PETR');
  if (v1=v2) or ((v1<=v3) and (v2<=v3))  then write(f_out,'RIVER');
  close(f_out);
end.
```

Задача 2. «Сеть»

- Для проведения олимпиады организаторы планируют объединить компьютеры участников в сеть. Из сетевого оборудования в наличии есть N коммутаторов и неограниченное количество сетевых кабелей. Коммутатор с номером i ($1 \le i \le n$) характеризуется числом a_i — количеством портов в этом коммутаторе.
- Организаторы могут соединить кабелем либо два коммутатора, либо два компьютера, либо коммутатор и компьютер. Каждый коммутатор может быть соединен кабелями не более чем с a_i устройствами (коммутаторами или компьютерами), каждый компьютер — не более чем с одним.

Задача 2. «Сеть»

- Два компьютера могут обмениваться данными, если от одного из них до другого можно добраться по кабелям, возможно, пройдя при этом по цепочке из коммутаторов. Организаторы хотят построить сеть таким образом, чтобы каждые два компьютера могли обмениваться данными.
- Какое максимальное количество компьютеров организаторы могут объединить в сеть, используя имеющиеся коммутаторы?

Задача 2. «Сеть»

- Входной файл **network.in**:

В первой строке входного файла находится одно число ***N*** — количество коммутаторов, имеющихся у организаторов ($0 \le N \le 10^5$).

Во второй строке файла находится ***N*** чисел ***a_i*** — количество портов в коммутаторе с номером ***i*** ($1 \le a_i \le 10^9$, $1 \le i \le N$).

- Выходной файл **network.out**:

Выводится единственное число — максимальное количество компьютеров, которое удастся объединить в сеть, используя имеющиеся коммутаторы.

Задача 1. «Сеть»

Примеры:

network.in	network.out
3 10 4 5	15
2 1 10	10
2 3 10	11

Задача 2. «Сеть»

- Обратим внимание на то, что коммутатор с одним портом бесполезен и, более того, вреден – он занимает порт у другого коммутатора, к которому мог быть подключен компьютер.
- Точно также бесполезен (хотя и не вредит!) коммутатор с двумя портами.
- Таким образом все коммутаторы, у которых один или два порта, необходимо исключить из рассмотрения.
- Найдем количество коммутаторов ***m***, у которых портов больше 2, а также общее количество портов у всех таких коммутаторов ***sp***.

Задача 2. «Сеть»

- По окончании цикла проверим ***m***.
- Если ***m=0***, т.е. не задействован ни один коммутатор, то соединить можно только два компьютера.
- В противном случае, для объединения в сеть ***m*** коммутаторов необходимо ***m-1*** соединение, т.е. ***2(m-1)*** порт. Таким образом, искомое количество компьютеров равно ***sp - 2(m-1)***.

Задача 2. «Сеть»

```
var
   n, i, p, m : longint;
   k, sp : int64;
   f_in, f_out : text;
begin
   assign(f_in,'network.in');  reset(f_in);
   readln(f_in,n);
   sp:=0; m:=0;
   for i:=1 to n do
   begin
```

Задача 2. «Сеть»

```
      read(f_in, p);
      if p>2 then begin
                  sp:=sp+p;   inc(m);
                        end;
   end;
   close(f_in);
   if m=0 then k:=2   else k:=sp-2*(m-1);
   assign(f_out,'network.out');   rewrite(f_out);
   write(f_out,k);   close(f_out);
end.
```

Задача 3. «Большое число »

- Дано целое число ***N***, состоящее из четного количества десятичных цифр.
  Над ним последовательно производятся следующие действия:

1. цифры числа разделяются на две равные половины;
2. левая и правая половины разворачиваются, то есть порядок следования цифр меняется на противоположный;
3. аналогичные действия выполняются для частей числа без первой и последней цифры, и так далее.

Задача 3. «Большое число »

- Когда останется последняя цифра первой половины числа и первая — второй, процесс останавливается, так как разворачивать их не имеет смысла.
- Рассмотрим пример. Пусть N = 1234567890. Тогда в процессе выполнения указанных действий будет получена следующая цепочка: 5432109876, 5123478906, 5143298706, 5142389706.
- Ваша задача — узнать результат последовательности указанных преобразований.

Задача 3. «Большое число »

- Входной файл **bignum.in**:

Входной файл содержит единственное число ***N*** (не более 10000 цифр). Допускаются нули в начале записи числа.

- Выходной файл **bignum.out**:

Выходной файл должен содержать единственное число — результат применения всех действий.

Задача 3. «Большое число»

Примеры:

bignum.in	bignum.out
1234567890	5142389706
000123	000231
012039	201390

Задача 3. «Большое число»

- Т.к. исходное число может содержать в начале нули, то хранить его нужно в строковом виде. А т.к. в нем может быть до 10000 цифр, то будем использовать тип AnsiString.
- Пусть
  n – количество цифр в числе;
  np – количество цифр в половине числа;

k – номер цифры в 1-й половине числа, с которой должна начаться перестановка цифр.

- Тогда

np-k+1 - количество переставляемых цифр в одной половине;

(np-k+1) div 2 - количество перестановок.

Задача 3. «Большое число»

- Пусть ***i*** – номер перестановки (***1≤ i ≤ (np-k+1) div 2***), тогда в 1-й половине переставляются цифры на позициях ***k+i-1*** и ***np-i+1***, а во 2-й половине – на позициях ***np+i*** и ***n-k-i+2***.
- В начале ***k=1***, затем, после выполнения всех перестановок для этого значения ***k***, оно увеличивается на 1.
- Данный цикл перестановок продолжается до тех пор, пока ***k*** не сравняется с ***np***.

Задача 3. «Большое число»

```
var
   n, i,  k, np : longint;
   d : AnsiString;
   s : char;
   f_in, f_out : text;
begin
   assign(f_in, 'bignum.in');  reset(f_in);
   readln(f_in,d);  close(f_in);
   n:=length(d);
   np:=n div 2;
```

Задача 3. «Большое число»

```
k:=1;
while k<>np do
begin
   for i:=1 to (np-k+1) div 2 do
   begin
      s:=d[k+i-1]; d[k+i-1]:=d[np-i+1]; d[np-i+1]:=s;
      s:=d[np+i]; d[np+i]:=d[n-k-i+2]; d[n-k-i+2]:=s;
   end;
   inc(k);
end;
```

Задача 3. «Большое число»

```
assign(f_out,'bignum.out');
   rewrite(f_out);
   write(f_out,d);
   close(f_out);
end.
```

Задача 4. «Ох, уж эти скобки»

- Математическое выражение записано в виде произведения:

$(\pm a_2x^2\pm a_1x\pm a_0)\cdot(\pm b_2x^2\pm b_1x\pm b_0)\cdot(\pm c_2x^2\pm c_1x\pm c_0)\cdot\ldots$

- Внутри каждой из ***N*** скобок произведения находится выражение вида:

$\pm a_2x^2\pm a_1x\pm a_0$,

где хотя бы один из коэффициентов a_i не равен нулю (b_i, c_i и т.д., аналогично).

Задача 4. «Ох, уж эти скобки»

- Требуется составить программу, которая перемножает выражения в скобках и выводит полученную функцию в виде многочлена с приведенными по степеням x слагаемыми, то есть в виде:

$$\pm q_{2N}x^{2N}\pm q_{2N-1}x^{2N-1}\pm \ldots\pm q_3x^3\pm q_2x^2\pm q_1x\pm q_0.$$

Задача 4. «Ох, уж эти скобки»

- Входной файл **brackets.in**:

В первой строке входного файла находится число ***N*** ($1 \le N \le 6$).

Во второй строке находится выражение из N пар скобок. Внутри каждой пары скобок находится выражение в виде «$\pm a_2 x\text{^}2 \pm a_1 x \pm a_0$», где «±»— это или знак «+», или знак «−».

Значение каждого из коэффициентов a_i, b_i, c_i и т.д. не превышает 10.

Задача 4. «Ох, уж эти скобки»

Выражение составлено по следующим правилам:

1. Всё выражение записывается, начиная со старшей степени переменной по убыванию степеней.
2. Если какой-то коэффициент равен нулю, то этот коэффициент и соответствующий ему x опускаются в записи вместе с арифметическим знаком. Исключением является случай, когда все коэффициенты равны нулю. В этом случае вместо всего выражения указывается единственный коэффициент 0.
3. Если $a_i=\pm1$ и $i>0$, то единица перед соответствующим ему x не ставится.
4. Если первый отличный от нуля коэффициент положителен, то знак «+» перед ним опускается.
5. В выражении отсутствуют пробельные символы (пробел, табуляция) и знаки умножения.

Задача 4. «Ох, уж эти скобки»

- Выходной файл **brackets.in**:

В первой строке выходного файла выводится результат раскрытия скобок в исходном выражении в следующем формате:

$\pm q_{2N} x\text{^}(2N) \pm q_{2N-1} x\text{^}(2N-1) \pm \ldots \pm q_1 x \pm q_0$.

Формат выражения должен полностью соответствовать описанию для входного файла. Скобки вокруг степеней ставить не нужно, они приведены здесь только для читабельности.

Задача 4. «Ох, уж эти скобки»

Примеры:

brackets.in	**brackets.out**
1 (3x^2+2x-1)	3x^2+2x-1
2 (4x^2+3x+5)(2x^2+4x+1)	8x^4+22x^3+26x^2+23x+5
3 (-x+7)(x)(x^2+x+1)	-x^4+6x^3+6x^2+7x
6 (1)(2)(3)(4)(5)(6)	720

Задача 4. «Ох, уж эти скобки»

- В массиве r[1..3,1..13] будем записывать коэффициенты трех многочленов (в 1-й строке – 1-й множитель, во 2-й строке – 2-й множитель, в 3-й строке – произведение.
- Процедура Skobka записывает коэффициенты трехчлена одной скобки либо в 1-ю либо во 2-ю строки массива.
- Процедура Sol перемножает многочлен 1-й строки массива на трехчлен 2-й строки массива, при этом результат сначала получается в 3-й строке, а затем переносится в 1-ю строку.
- Процедура Output выводит полученный результат.

Задача 4. «Ох, уж эти скобки»

- В процедуре Skobka сначала в строке s ищется позиция p символов x^. Если p>0, то s[1] .. s[p-1] образуют коэффициент при x^2. При этом нужно учесть случаи, когда p=1 (x^2), p=2 и s[1]='-' (-x^2). Символы с 1 по p+2 в строке s удаляются.
- Затем в строке s ищется позиция p символа x. Если p>0, то s[1] .. s[p-1] образуют коэффициент при x. При этом нужно учесть случаи, когда p=1 (x), p=2 и s[1]='-' (-x), p=2 и s[1]='+' (+x).
  Символы с 1 по p в строке s удаляются.
- Оставшиеся символы в строке s образуют свободный член трехчлена.

Задача 4. «Ох, уж эти скобки»

x	x^{12}	x^{11}	x^{10}	x^{9}	x^{8}	x^{7}	x^{6}	x^{5}	x^{4}	x^{3}	x^{2}	x^{1}	x^{0}
i	1	2	3	4	5	6	7	8	9	10	11	12	13
a			a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
b											b_2	b_1	b_0
c	c_{12}	c_{11}	c_{10}	c_9	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0

- Рассмотрим принцип работы процедуры Sol на примере нахождения коэффициента c_8:

$$c_8x^8=a_8x^8\cdot b_0+a_7x^7\cdot b_1x+a_6x^6\cdot b_2x^2=(a_8b_0+a_7b_1+a_6b_2)\cdot x^8$$

$$C_8=a_8b_0+a_7b_1+a_6b_2$$

- Остальные коэффициенты находятся аналогично.

Задача 4. «Ох, уж эти скобки»

- При выводе результата необходимо учитывать следующие моменты;

1. Перед каждым положительным коэффициентом, кроме первого слагаемого, должен стоять знак +.
2. Коэффициент 1 не выводится.
3. У коэффициента -1 выводится только знак -.
4. Пункты 2 и 3 не распространяются на вывод свободного члена.
5. Если все коэффициенты результата равны 0, то необходимо вывести 0.

Задача 4. «Ох, уж эти скобки»

```
var
   n, i, p : longint;
   r : array[1..3,1..13] of longint;
   s : String;
   f_in, f_out : text;
procedure skobka(s:string;k:integer);
var
   p, i, t : integer;
begin
   for i:=1 to 13 do r[k,i]:=0;
```

Задача 4. «Ох, уж эти скобки»

```
p:=pos('x^',s);
if p>0
then begin
          if p=1
          then r[k,11]:=1
          else if (p=2) and (s[1]='-')
                    then r[k,11]:=-1
                    else val(copy(s,1,p-1),r[k,11],t);
          delete(s,1,p+2);
        end;
p:=pos('x',s);
```

Задача 4. «Ох, уж эти скобки»

```
if p>0
 then begin
            if p=1
            then r[k,12]:=1
            else if (p=2) and (s[1]='+')
                    then r[k,12]:=1
                    else if (p=2) and (s[1]='-')
                            then r[k,12]:=-1
                            else val(copy(s,1,p-1),r[k,12],t);
            delete(s,1,p);
          end;
  val(s,r[k,13],t);
end;
```

Задача 4. «Ох, уж эти скобки»

```
procedure sol;
var i,j:integer;
begin
   for i:=1 to 13 do r[3,i]:=0;
   for i:=2 downto 0 do
      for j:=10 downto 0 do
         r[3,13-i-j]:=r[3,13-i-j]+r[1,13-j]*r[2,13-i];
   for i:=1 to 13 do r[1,i]:=r[3,i];
end;
```

Задача 4. «Ох, уж эти скобки»

```
procedure output;      var f:integer;
begin
   f:=0;
   for i:=1 to 13 do
     if r[1,i]<>0
     then begin
               if (f=1) and (r[1,i]>0) then write(f_out,'+');
               if r[1,i]=-1 then if i=13 then write(f_out,'-1‘) else write(f_out,'-‘)
                            else if (r[1,i]<>1) or (i=13) then write(f_out,r[1,i]);
               if i<13 then write(f_out,'x');
               if i<12 then write(f_out,'^',13-i);
               f:=1;
            end
   if f=0 then write(f_out,0);
end;
```

Задача 4. «Ох, уж эти скобки»

```
begin
   assign(f_in,'brackets1.in');  reset(f_in);
   readln(f_in,n);   readln(f_in,s);  close(f_in);
   if n=1
   then skobka(copy(s,2,length(s)-2),1)
   else begin
            p:=pos(')',s); skobka(copy(s,2,p-2),1);   delete(s,1,p);
            for i:=2 to n do
            begin
               p:=pos(')',s);  skobka(copy(s,2,p-2),2); delete(s,1,p);
               sol;
            end;
        end;
```

Задача 4. «Ох, уж эти скобки»

```
    assign(f_out,'brackets.out');
    rewrite(f_out);
    output;
    close(f_out);
end.
```

Задача 5. «Разбиение числа »

- Факториалом числа n называется произведение $n!=1\cdot2\cdot\ldots\cdot n$ при $n>0$ и $n!=1$ при $n=0$.
- Количество сочетаний из n элементов по k определяется следующим образом:

$C_n^k=n!/(k!(n-k)!)$, если $0\le k\le n$, и $C_n^k=0$, если $k>n$.

- В математике такие числа называются также биномиальными коэффициентами.
- Требуется представить заданное число ***P*** в виде суммы трех биномиальных коэффициентов:

$P=C_a^1+C_b^2+C_c^3,\ 0\le a<b<c$

Задача 5. «Разбиение числа»

- Входной файл **decomp.in**:

Входной файл содержит единственное число ***P*** ($1 \le P \le 10^{18}$).

- Формат выходного файла **decomp.out**:
- В выходной файл выводится искомые числа a, b, c ($0 \le a < b < c$), разделённые пробелами. Если задача не имеет решения, выведите три нуля.
- Примеры:

decomp.in	decomp.out
42	1 4 7
31	1 5 6

Задача 5. «Разбиение числа»

- Сначала определим, как вычислять значения C_a^1, C_b^2 и C_c^3:

$C_a^1 = a!/(1!\,(a-1)!) = a$,

$C_b^2 = b!/(2!(b-2)!) = b(b-1)/2$,

$C_c^3 = c!/(3!(c-3)!) = c(c-1)(c-2)/6$.

- Найдем наибольшее значение ***c***, для которого значение C_c^3 меньше введенного числа ***p***.
  Для этого используем функцию **fc**, реализующую бинарный поиск в диапазоне от 2 до 2000000.
- Аналогично в диапазоне от 1 до 2000000 ищем наибольшее значение ***b***, для которого $C_b^2 \le p - C_c^3$ (функция **fb**).
- Находим $a = p - C_c^3 - C_b^2$.

Задача 5. «Разбиение числа»

```
var
    p, s, n, i, a, b, c : int64;
    f_in, f_out : text;
function fc(x:int64):int64;
var  min, max, s : int64;
begin
    min:=2; max:=2000000;
    while max-min>1 do
    begin
        s:=(max+min) div 2;
        if s*(s-1)*(s-2) div 6>x  then max:=s else min:=s
    end;
    fc:=min;
end;
```

Задача 5. «Разбиение числа»

```
function fb(x:int64):int64;
var min, max, s : int64;
begin
    min:=1; max:=2000000;
    while max-min>1 do
    begin
        s:=(max+min) div 2;
        if s*(s-1) div 2>x  then max:=s else min:=s
    end;
    fb:=min;
end;
```

Задача 5. «Разбиение числа»

```
begin
    assign(f_in,'decomp.in');  reset(f_in);
    readln(f_in,p);  close(f_in);
    c:=fc(p);
    p:=p-c*(c-1)*(c-2) div 6;
    b:=fb(p);
    a:=p-b*(b-1) div 2;
    assign(f_out,'decomp.out');  rewrite(f_out);
    write(f_out,a,' ',b,' ',c);;
    close(f_out);
end.
```

Задача 6. «НОК»

- Наименьшим общим кратным (НОК) нескольких чисел называется наименьшее натуральное число, которое делится на каждое из этих чисел.
- Заданы два числа ***N*** и ***K***. Требуется найти набор из ***N*** различных натуральных чисел, наименьшее общее кратное которых равняется ***K***.
- Среди всех этих чисел не должно быть единицы и самого числа ***K***.

Задача 6. «НОК»

- Входной файл **lcm.in**:

В первой строке входного файла записаны через пробел два числа N и K ($1 \le N \le 1000$, $1 \le K \le 10^9$).

- Выходной файл **lcm.out**:

В выходной файл в первой строке выводится искомый набор из N чисел, разделённых пробелами.
Если Вы смогли найти несколько наборов, то выведите любой из них.
Если требуемого набора не существует, тогда выведите -1.

Задача 6. «НОК»

Примеры:

lcm.in	lcm.out
2 14	2 7
12 20736	3 9 27 81 256 128 64 32 16 8 4 2
17 42	-1
7 123456	2 3 4 6 30864 41152 61728

Задача 6. «НОК»

- На 1-м этапе исходное число ***K*** разложим на простые множители: $K=p_1^{\alpha 1}\cdot p_2^{\alpha 2}\cdot\ldots\cdot p_t^{\alpha t}$.
- При этом простые делители $p_1, p_2, \ldots, p_t$ будем хранить в массиве ***P***, а количество таких делителей $\alpha_1, \alpha_2, \ldots, \alpha_t$ – в массиве ***KP***.
- Определим общее количество делителей:

  $$r=(\alpha_1+1)(\alpha_2+1)\ldots(\alpha_t+1)-2$$

- Если окажется, что $r<n$ (или $t<2$ или $n=1$), то выводим ***-1***.
- В противном случае переходим ко 2-му этапу.

Задача 6. «НОК»

- Вычислим и выведем два числа: $d_1=p_t^{\alpha t}$ (произведение всех наибольших простых делителей) и $d_2=K \; div \; p_1$.
- В вспомогательный массив ***a*** запишем все степени первого простого делителя p_1. При этом запомним позицию в массиве, где закончились эти числа.
- Все степени второго простого делителя p_2 также запишем в массив ***a***. Кроме того, туда же запишем все произведения этих степеней на все числа, ранее записанные в массив. При этом снова запомним конечную позицию в массиве.
- Эти же действия повторим для всех остальных простых делителей.

Задача 6. «НОК»

- Во время выполнения действий 2-го этапа нужно выполнять следующее:

1. Все получаемые числа, если они не совпадают с d_1 и d_2, выводим в файл.
2. Ведем подсчет количества выведенных чисел, как только это количество совпадет с ***N***, работу программы завершаем.

Задача 6. «НОК»

```
const
    maxp=1000000;
var
    n, i, k, m, x, t, j, r, q, d, qq, l, d1, d2, kd : longint;
    p, kp : array[1..maxp] of longint;
    a : array[1..1000] of int64;
    f_in,f_out:text;
begin
   assign(f_in,'lcm2.in');  reset(f_in);
   read ln(f_in,n,k);  close(f_in);
   for i:=1 to maxp do kp[i]:=0;
```

Задача 6. «НОК»

```
x:=k;
while x mod 2=0 do
begin
    inc(kp[1]); x:=x div 2;
end;
if kp[1]>0
then begin
            p[1]:=2;
            t:=2;
        end
else t:=1;
```

Задача 6. «НОК»

```
m:=3;
repeat
   while x mod m=0 do
   begin
       inc(kp[t]); x:=x div m;
   end;
   if kp[t]>0
   then begin
                p[t]:=m;
                inc(t);
            end;
   inc(m,2);
until x=1;
dec(t);
```

Задача 6. «НОК»

```
r:=1;   for i:=1 to t do r:=r*(kp[i]+1);
r:=r-2;
assign(f_out,'lcm.out');  rewrite(f_out);
if (t<2) or (r<n) or (n=1)
then write(f_out,-1)
else begin
        d1:=1;  for j:=1 to kp[t] do d1:=d1*p[i];
        d2:=k div d1;
        write(f_out,d1,' ',d2,' ');
        kd:=2;
        if kd=n then begin
                        close(f_out);  halt
                     end;
```

Задача 6. «НОК»

```
q:=0;
for i:=1 to t do
begin
    d:=1;
    for j:=1 to kp[i] do
    begin
        d:=d*p[i];   inc(q);    a[q]:=d;
        if (d<>d1) and (d<>d2)
        then begin
                write(f_out,d,' ');   inc(kd);
                if kd=n then begin
                                close(f_out);  halt
                             end;
             end
    end;
```

Задача 6. «НОК»

```
if i>1 then
begin
      for l:=1 to qq do
      begin
           d:=a[l];
           for j:=1 to kp[i] do
           begin
              d:=d*p[i];   inc(q);     a[q]:=d;
              if (d<>d1) and (d<>d2)
              then begin
                        write(f_out,d,' ');
                        inc(kd);
```

Задача 6. «НОК»

```
                    if kd=n then begin
                                    close(f_out);  halt
                                  end;
                  end
                end;
              end;
            end;
            qq:=q;
          end;
      end;
      close(f_out);
end.
```