ВСЕРОССИЙСКАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ПО ИНФОРМАТИКЕ

школьный этап (программирование)

в 2012-2013 учебном году

Цель олимпиады по информатике — **способствовать поиску наиболее одаренных школьников** .

Важной особенностью задач, используемых при проведении школьного и муниципального этапов, является ориентация их на проверку развития у учащихся теоретического мышления, логики, а также творческих способностей и интуиции.

Задачи школьного этапа олимпиады должны быть такой сложности, чтобы не отпугнуть учащихся, а дать им возможность продемонстрировать свои лучшие качества.

Основные критерии отбора олимпиадных задач для проведения школьного и муниципального этапов Всероссийской олимпиады школьников по информатике :

- оригинальная формулировка задачи (или идея ее решения);
- в тексте условия задачи не должны встречаться термины и понятия, выходящие за пределы изучаемых в рамках базового учебного плана предметов;
- задача должна быть однозначно определена;
- задача не должна требовать для своего решения специальных знаний;
- формулировка задачи должна предполагать наличие этапа формализации при ее решении;
- задача должна быть разумной сложности и трудоемкости.

Из опыта олимпиад можно выделить наиболее часто встречающиеся разделы информатики, к которым с можно отнести тематику задач:

- комбинаторика;
- сортировка и поиск;
- обработка последовательностей;
- алгоритмы на графах;
- элементы вычислительной геометрии.
- перебор вариантов и методы его сокращения;
- динамическое программирование;

Методика решения олимпиадных задач

Этапы решения олимпиадных задач:

- Разбор условия задачи.
- Формализация условия задачи.
- Разработка алгоритма решения задачи.
- Программная реализация алгоритма.
- Отладка и тестирование программы.
- Отправка решения на проверку.

Важно отметить, что **текст задачи нужно всегда внимательно читать** от начала и до конца, поскольку ключевое условие может быть спрятано, например, в формате входных или выходных данных, а также в приведенных примерах файлов входных и выходных данных.

При разработке программы следует также обратить особое внимание на описание **формата входных и выходных данных**, приведенное в условии задачи. Имена входного и выходного файлов также описаны в условии задачи, и неправильное их написание в программе считается ошибкой.

Необходимо помнить при написании программы, — это **сохранение редактируемых файлов** во время тура.

Полученная программа должна соответствовать заданной **размерности входных данных** и удовлетворять ограничениям на **память и время работы**, заданные в условии задачи.

Задача 1.

Напечатать все трехзначные десятичные числа, сумма цифр которых равна данному числу.

Один из вариантов решения перебором

```
var a,b,c,n,k:integer;
begin
write('n=');  readln (n);
for a:=1 to 9 do
  for b:=0 to 9 do
    for c:=0 to 9 do
        if a+b+c=n then
                begin
                writeln (a,b,c,'   ');
                k:=k+1;
                end;
  writeln;
 writeln ('k=',k) ;
  writeln;
end.
```

Второй вариантов решения перебором

```
Var  a,b,c,n,k,m: integer;
begin
write('n='); readln(n);
  for m:=100 to 999 do
        begin
         c:=m mod 10;
         b:= m div 10 mod 10;
         a:= m div 100;
          if a+b+c=n then
                begin
                 write(m:5);
                  k:=k+1;
                end;
          end;
    writeln('k=',k)
end.
```

Задача 2. «Малыш и Карлсон».
Малыш и Карлсон живут в прямоугольной комнате размером ***A x B*** . Как им посчитать, сколько понадобится квадратных ковриков со стороной ***C***, чтобы полностью покрыть пол комнаты? (Малыш и Карлсон не умеют ни делить, ни умножать.) Напишите программу для решения этой задачи.

Алгоритм решения:

во внешнем цикле по одной из сторон комнаты (***while p<a***) резервируем место для ряда (***p:=p+c***), затем во внутреннем цикле по другой стороне (***while m<b***) проверяем, сколькими ковриками можно закрыть ряд, оператор ***m:=m+c*** резервирует место для коврика, а оператор ***kovrik:=kovrik+1*** подсчитывает общее количество уложенных ковриков.

Программа на языке программирования Паскаль.

```
var   a, b, c, kovrik, m, p: integer;
begin
readln(a, b, c);
kovrik:= 0;
p:=  0;
while p < a do
    begin
    p:= p +  c;
    m:=  0;
     while m < b do
          begin
          m:= m + c;
          kovrik:= kovrik + 1
          end
     end;
writeln(kovrik)
end.
```

Задача 3. «Бактерии».

Колония состояла из n бактерий (не более 30000). В нее попал вирус, который в первую минуту уничтожил одну бактерию, а затем разделился на два новых вируса. Одновременно каждая из оставшихся бактерий тоже разделилась на две новые. В следующую минуту возникшие два вируса уничтожили две бактерии, а затем все вирусы и бактерии снова разделились и так далее. Будет ли эта колония жить бесконечно долго или вымрет?

Ваша программа должна:

- Запросить число бактерий n;
- Выяснить и сообщить, через сколько суток, часов и минут колония бактерий прекратит свое существование или выдать сообщение, что колония вечна.

Пример ответа: Для n=A. Ответ – B суток C часов D минут (где A, B, C, D – числовые значения).

Программа на языке программирования Паскаль.

```
Var     a, b, c:   shortint;
          t, n, v:   longint;
begin
Write (‘Начальная численность колонии  -');     readln ( n);
v:=1;
while n>0 do
      begin
                  t:= t + 1;                  { минуты}
                  n:= ( n - v ) * 2;          { бактерии}
                  v:= v * 2;                  { вирусы}
               end;
a:= t div 1440;
b:= ( t – a * 1440) div 60;
c:= t – a - b;
Write (‘Колония  прекратит существование через  ',a, ' суток  ’,
b,‘  часов  ', c , ’   минут');
end.
```

Задача 4.

Дан прямоугольник со сторонами A и B, где A, B - натуральные числа. Начинаем отсекать от него квадраты (рис.1). Сколько таких квадратов можно отсечь, если каждый раз отсекается самый большой квадрат?

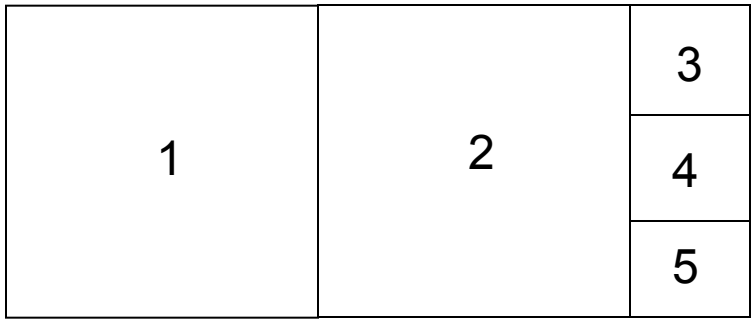

Рис. 1. Отсечение квадратов

1 способ.

Для решения этой задачи нам нужны функции MAX и MIN, для их определения используем подпрограммы-функции.

Введем:

- вспомогательные переменные X и Y (Y >= X), соответствующие уменьшающимся сторонам прямоугольника;
- вспомогательную переменную D, которая определяет уменьшение размеров прямоугольника после очередного отсечения наибольшего квадрата, сторона которого находится как X := MIN(D, X).

Организуем цикл, в котором сторона Y уменьшается каждый раз на MIN(D, X) до тех пор, пока не останется последний квадрат или Y не станет меньше X. В последнем случае переименовываем стороны оставшегося прямоугольника как Y := MAX(D, X) и X := MIN(D, X) и продолжаем цикл.

```
var  a, b, d, k, x, y: integer;
function min ( i, j: integer): integer;
begin
      if  i < j   then   min:=i      else min:=j
end;
function max ( i, j : integer): integer;
begin
     If  i < j  then  max:=j      else max:=i
end;
begin
    repeat
    Writeln ('vvedite dva naturalnix chisla');       readln (a, b);
     until (a>0) and (b>0);
  k:=1;
  x:=min(a,b);
  y:=max(a,b);
   while  x <> y  do   begin
                              k:=k+1;
                              d:=y-x;
                              y:=max(d,x);
                              x:=min(d,x);
                             end;
Writeln ( 'iskomoe chislo kvadratov:', k )
end.
```

2 способ.

Задачу можно решить с помощью стандартных функций PASCAL: ***Y DIV X*** и ***Y MOD X,*** используя алгоритм Евклида.

Алгоритм решения:

Организуем цикл, в котором формируем остатки от деления $r_0, r_1, r_2, \ldots, r_n, r_{n+1}$ до тех пор, пока один из этих остатков не станет равен нулю $r_{n+i} = 0$. Таким образом, мы строим функцию порождения остатка от деления $r_{n+i} = r_n \bmod r_{n-i}$, где $r_0 = A$ и $r_i = B$. Для той же самой системы остатков мы можем посчитать, сколько раз нацело укладывается остаток r_{n-i} в r_n.

,

```
{алгоритм Евклида}
VAR A,B,R0,R,R1, K:INTEGER;
BEGIN
  REPEAT
        WRITE('ВВЕДИТЕ НАТУРАЛЬНОЕ ЧИСЛО   A = ');
        READLN(A);
        WRITE('ВВЕДИТЕ НАТУРАЛЬНОЕ ЧИСЛО B<= A, B = ');
        READLN(B);
   UNTIL (B > 0) AND (A > 0) AND (A >=B);
 R0 := A; R1 := B;
 К := R0 DIV R1;
  WHILE R0 MOD R1 <> 0 DO   BEGIN
                                        R := R0 MOD R1;
                                        R0 := R1;
                                        R1 := R;
                                        К := К + R0 DIV R1
                                       END;
WRITELN('ИСКОМОЕ ЧИСЛО КВАДРАТОВ  К = ',К);
END.
```