ЕГЭ по информатике (часть С)

Павлова Елена Станиславна

es.pavlova@yandex.ru

старший преподаватель кафедры ВТ ВолгГТУ, председатель экспертной комиссии ЕГЭ по информатике Волгоградской области

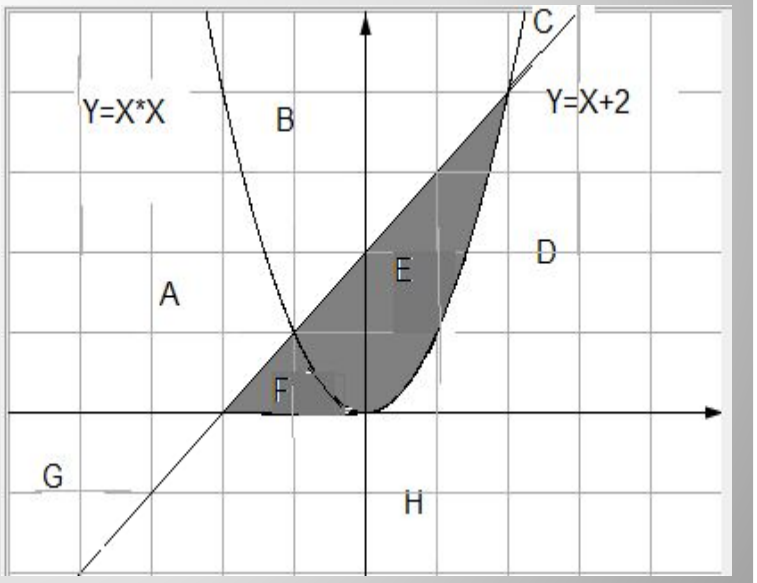
Репетиционное ЕГЭ по информатике

(1 вариант)

Задача С1

Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y – действительные числа) и определяется принадлежность этой точки заданной закрашенной области (включая границы). Программист торопился и написал программу неправильно.

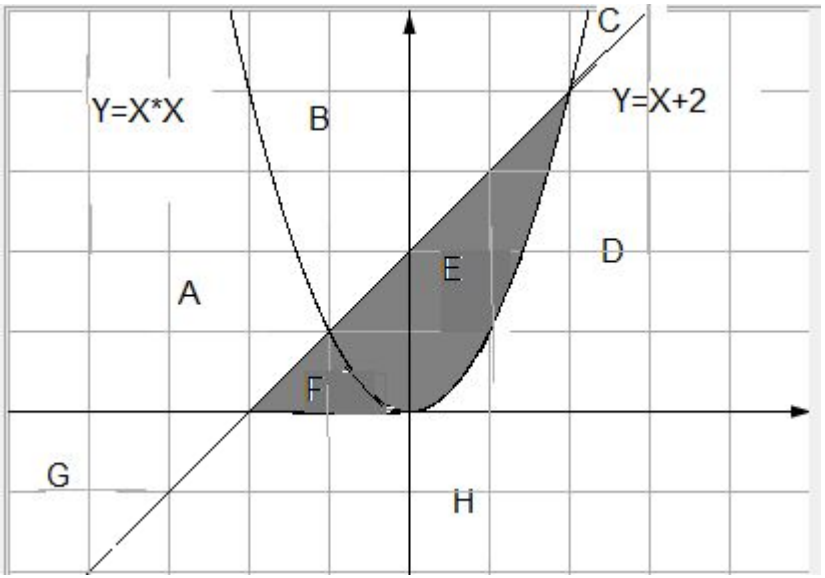
```
var x,y: real;
begin
readln(x,y);
 if y>=0 then
   if y<=x+2 then
      if y>=x*x then
      write('принадлежит')
    else
      write('не принадлежит')
end.
```



1) Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F, G и H). Точки, лежащие на границах областей, отдельно не рассматривать.

   В столбцах условий укажите "да", если условие выполнится, "нет" если условие не выполнится, "—" (прочерк), если условие не будет проверяться, «не изв.», если программа ведет себя по-разному для разных значений, принадлежащих данной области. В столбце "Программа выведет" укажите, что программа выведет на экран. Если программа ничего не выводит, напишите "—" (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «не изв». В последнем столбце укажите "да" или "нет".

2) Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, достаточно указать любой способ доработки исходной программы.)




```
var x,y: real;
begin
readln(x,y);
 if y>=0 then
   if y<=x+2 then
      if y>=x*x then
      write('принадлежит')
    else
      write('не принадлежит')
end.
```

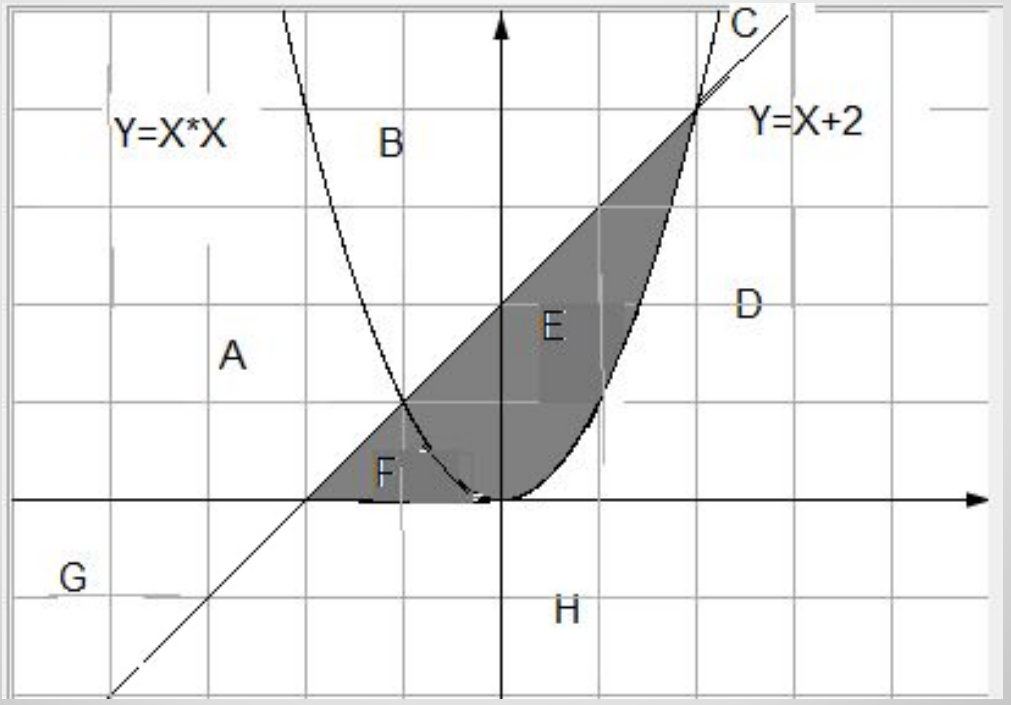
Область	y>=0?	y<=x+2	y>=x*x?	вывод	верно?
A					
B					
C					
D					
E					
F					
G					
H					

Область	y>=0?	y<=x+2	y>=x*x?	вывод	верно?
A	да	нет	-	-	нет
B	да	нет	-	-	нет
C	да	нет	-	-	нет
D	да	да	нет	не принадлежит	да
E	да	да	да	принадлежит	да
F	да	да	нет	не принадлежит	нет
G	нет	-	-	-	нет
H	нет	-	-	-	нет

Условия для заштрихованных областей:

Область E: (y<=x+2) and (y>=x*x)

Область F: ((y<=x+2) and (y<=x*x) and (y>=0)and (x<=0))



Поэтому часть программы после доработки может быть следующего вида:

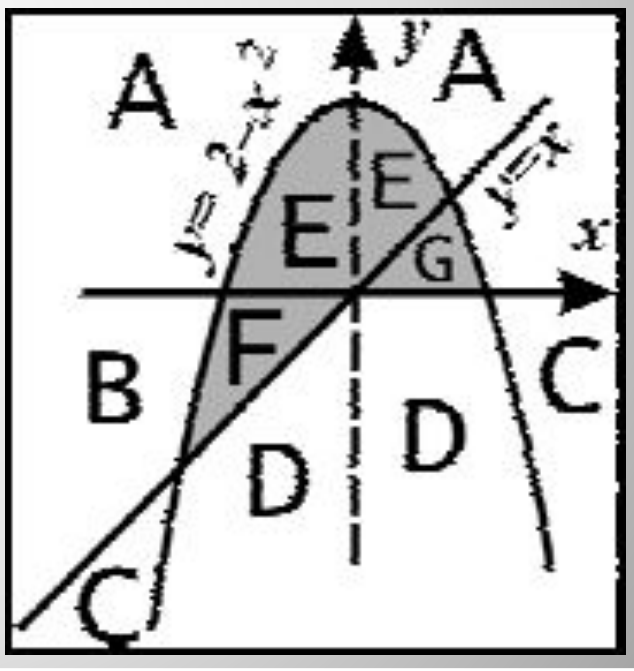
```
If ((y<=x+2) and (y>=x*x)) or (
  (y<=x+2) and (y<x*x) and (y>=0)
  and (x<=0)) then
  write('принадлежит')
 else
     write('не принадлежит');
```

Или после упрощения логического выражения:

```
If (y<=x+2) and ((y>=x*x) or (
  (y<x*x) and (y>=0) and (x<=0)))
  then
  write('принадлежит')
 else
   write('не принадлежит');
```

*Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (**x**, **y** – действительные числа) и определяет принадлежность точки заштрихованной области, включая ее границы. Программист торопился и написал программу неправильно. Вот она:*

```
var x,y: real;
begin
readln(x,y);
if y>=x then
  if y>=0 then
    if y<=2-x*x then
      write('принадлежит')
    else
      write('не принадлежит')
end.
```



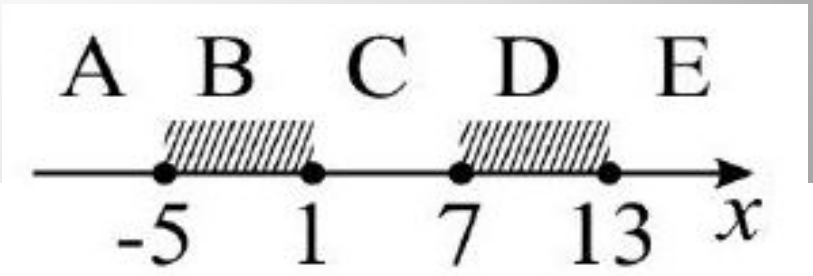
Область	y>=x?	y>=0?	y<=2-x*x?	вывод	верно?
A					
B					
C					
D					
E					
F					
G					

Область	y>=x?	y>=0?	y<=2-x*x?	вывод	верно?
A	да	да	нет	не принадлежит	да
B	да	нет	–	–	нет
C	нет	–	–	–	нет
D	нет	–	–	–	нет
E	да	да	да	принадлежит	да
F	да	нет	–	–	нет
G	нет	–	–	–	нет

```
var x,y: real;
begin
readln(x,y);
if (y <=2-x*x) and ((y>=x) or (y>=0)) then
  write('принадлежит')
else
  write('не принадлежит')
end.
```

Требовалось написать программу, при выполнении которой с клавиатуры считывается координата точки на прямой (*x* – действительное число) и определяется принадлежность этой точки одному из выделенных отрезков B и D (включая границы). Программист торопился и написал программу неправильно.

```
var x: real;
begin
  readln(x) ;
  if x>l then
    if x>=7 then
      if x>13 then
        write('не принадлежит')
      else
        write('принадлежит')
end.
```



Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D и E).
Границы (точки -5, 1, 7 и 13) принадлежат заштрихованным областям.

Область	Условие 1 (x>1)	Условие 2 (x>=7)	Условие 3 (x>13)	Программа выведет	Область обрабатывается верно
A					
B					
C					
D					
E					

Задача С2

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от 0 до 10000. Опишите на русском языке или на одном из языков программирования алгоритм, который позволяет найти и вывести произведение элементов массива, которые имеют двузначное значение и не оканчиваются на 2. Гарантируется, что в исходном массиве есть хотя бы один элемент, значение которого является двузначным числом, и при этом его последняя цифра не равна двум.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из них.

```
Const
   N=30;
Var
  a: array [1..N] of integer;
   i, j, p: longint;
begin
  for i:=1 to N do
      readln(a[i]);
 ...
end.
```

В качестве ответа вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например, *Borland Pascal 7.0*) или в виде блок-схемы. В этом случае вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

```
P:=1;
For i:=1 to n do
  if (a[i]>9) and (a[i]<100) and (a[i] mod
  10 <> 2) then
    P:=P*a[i];
Writeln (P);
```

Задача С3

Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу **один камень** или увеличить количество камней в куче в **два раза**. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16 или 30 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней.

Игра завершается в тот момент, когда количество камней в куче становится не менее 34. Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 34 или больше камней.

В начальный момент в куче было S камней, $1 \le S \le 33$.

Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника.

Выполните следующие задания. Во всех случаях обосновывайте свой ответ.

1. а) Укажите все такие значения числа S, при которых Петя может выиграть в один ход. Обоснуйте, что найдены все нужные значения S, и укажите выигрывающий ход для каждого указанного значения S.

Последним ходом может быть «+1» или «*2». Выиграть последним ходом «+1» можно, если S = 33.

Ходом «*2» можно выиграть из любой позиции при S >= 17 (или S>16) (сюда входит и 33!).

Ответ для 1а. *Петя может выиграть за один ход при любом S > 16. Он должен увеличить вдвое число камней, при этом в куче всегда получится не менее 34 камней.*

1. б) Укажите такое значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Опишите выигрышную стратегию Вани.

Для ответа на этот вопрос нужно найти позицию, из которой все возможные ходы ведут к выигрышу за 1 ход, то есть к позициям, найденным в пункте 1а.

Ответ для 1б: При S = 16 Петя не может выиграть в один ход, потому что при его ходе «+1» число камней в куче становится равно 17 (меньше 34), а при ходе «*2» число камней в куче становится равно 32 (также меньше 34). Других возможных ходов у Пети нет.

Из любой позиции после одного хода Пети (это может быть 17 или 32), Ваня может выиграть своим первых ходом, удвоив количество камней в куче.

2. Укажите два таких значения S, при которых у Пети есть выигрышная стратегия, причём

– Петя не может выиграть за один ход,
– Петя может выиграть своим вторым ходом, независимо от того, как будет ходить Ваня.

Для каждого указанного значения S опишите выигрышную стратегию Пети.

Пете, для того, чтобы гарантированно выиграть на втором ходу, нужно из начальной позиции перевести игру в одну из проигрышных позиций, например, S = 16 (см п.1б). Петя может перевести игру в эту позицию из позиций S = 15 (ходом «+1») и S = 8 (ходом «*2»).

Ответ для п.2: из позиций S = 15 и S = 8 Петя не может выиграть в один ход, но Петя может выиграть своим вторым ходом, независимо от того, как будет ходить Ваня. При S = 15 ходом «+1» Пете нужно перевести игру в позицию S = 16, которая является проигрышной (см. ответ на вопрос 1б). При S = 8 Петя переводит игру в ту же позицию ходом «*2».

3. Укажите значение S, при котором:
– у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети, и
– у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

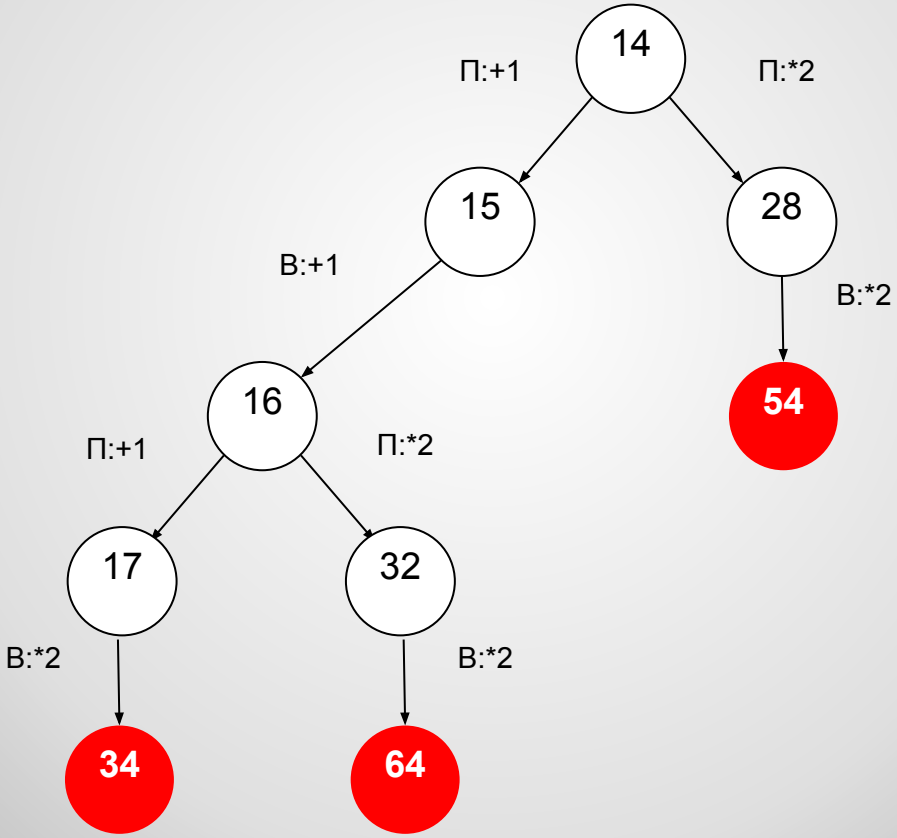
Для указанного значения S опишите выигрышную стратегию Вани.

Постройте дерево всех партий, возможных при этой выигрышной стратегии Вани (в виде рисунка или таблицы). На рёбрах дерева указывайте, кто делает ход, в узлах – количество камней в куче.

Нужно найти такую позицию, из которой оба возможных хода Пети ведут к проигрышу, т.е. в Петя не должен попасть в позицию, из которых возможны и выигрыш в 1 ход, и выигрыш в 2 хода. Например, это позиция S = 14, из которой можно «попасть» только в S = 15 («выигрыш в два хода») и S = 28 («выигрыш в один ход»).

Ответ для п.3: В позиции S = 14 у Вани есть выигрышная стратегия, которая позволяет ему выиграть первым или вторым ходом. Если Петя выбирает ход «+1», в куче становится 15 камней и Ваня выигрывает на 2-м ходу (см. ответ на вопрос 2). Если Петя выбирает ход «*2», Ваня выигрывает первым ходом, удвоив число камней в куче.

Остается нарисовать дерево возможных вариантов игры из позиции S = 14.



Обратите внимание, что на каждом шаге мы рассматриваем все возможные ходы Пети и только один лучший ход Вани.

Все ходы Вани мы не рассматриваем, потому что мы хотим доказать, что **у Вани** есть выигрышная стратегия – ему достаточно одного хода, после которого он выиграет.

В то же время нужно рассмотреть **все возможные ответы Пети**, чтобы доказать, что у него нет шансов на выигрыш при правильной игре Вани. В этом суть теории игр – добиться лучшего результата в худшем случае, то есть при безошибочной игре соперника.

Задача С4

На вход программе подается зашифрованный текст заклинания, состоящего не более, чем из 200 символов. Этот текст представляет собой последовательность слов (т.е. непрерывных последовательностей английских букв длиной не более 20 символов), разделенных произвольным количеством пробелов и знаков препинания («,», «.», «:», «-»), которая заканчивается точкой. Символы входной строки после точки, если они есть, к заклинанию не относятся и поэтому игнорируются. Заклинание было зашифровано Гарри Поттером следующим образом. Сначала Гарри определил количество букв в самом длинном слове, обозначив полученное число K. Затем он заменил каждую английскую букву в заклинании на букву, стоящую в алфавите на K букв далее (алфавит считается циклическим, то есть, после буквы Z стоит буква A), оставив другие символы неизменными. Строчные буквы при этом остались строчными, а прописные – прописными.

Требуется написать программу как можно более эффективную программу, которая будет выводить на экран текст расшифрованного заклинания заклинания. Например, если исходный текст был таким:

Ce Ud Fd Gde Ud.

то результат расшифровки должен быть следующий:

Zb Ra Ca Dab Ra.

Из условия следует, что задача решается в два этапа:

1) прочитать символы до точки и определить длину самого длинного слова из латинских букв (обозначим ее **maxLen**);

2) сделать «сдвиг» кодов латинских букв на **maxLen** влево.

Простое посимвольное чтение строки **s** до первой встреченной точки выглядит так (здесь **c** – переменная типа **char**):

```
s := ''; { пустая строка }
repeat
  read(c);    { прочитали символ }
  s := s + c; { добавили в конец
строки }
until c = '.';
```

При этом нам нужно еще определить длину самого длинного слова с учетом того, что между словами может быть сколько угодно символов-разделителей (разных!).
Введем переменную **len**, которая будет определять длину текущего (очередного, вводимого в данный момент) слова.

Как определить, что прочитанный символ – латинская буква? Можно использовать условный оператор со сложным условием:

```
if (('a' <= c) and (c <= 'z')) or
   (('A' <= c) and (c <= 'Z')) then ...
```

Или это можно сделать с помощью оператора in, который проверяет, входит ли элемент во множество:

```
if c in ['a'..'z', 'A'..'Z'] then ...
```

Здесь множество в квадратных скобках содержит два интервала: от 'a' до 'z' и от 'A' до 'Z'.

Если очередной прочитанный символ – латинская буква, нужно увеличить **len** на единицу (слово продолжается). Если же это не латинская буква, то слово закончилось, так как встречен символ-разделитель .

Полученное значение переменной **len** нужно сравнить с максимальной длиной и, если прочитанное слово длиннее всех предыдущих, записать его длину в **maxLen**. Таким образом, цикл ввода выглядит так:

```
s := '';
maxLen := 0;
len := 0;
repeat
    read(c);
    s := s + c;
    if c in['a'..'z','A'..'Z'] then
      len := len + 1
    else
       begin
           if len > maxLen then
              maxLen := len;
           len := 0;
       end;
until c = '.';
```

Теперь нужно в цикле пройти всю прочитанную строку и «сдвинуть» каждый символ (точнее, его код) вправо на **maxLen**:

```
for i:=1 to Length(s) do
  if s[i] in ['a'..'z','A'..'Z'] then
    begin
      code := Ord(s[i]);  { старый код }
      newcode := code - maxLen;
      { новый код }
      s[i] := Chr(newcode);
    end;
```

Однако такое решение не учитывает цикличность: например, при сдвиге буквы **'A'** на 2 символа влево мы не получим **'Y'**. Поэтому после изменения кода нужно проверить, не вышел ли он за допустимые границы (диапазона латинских букв), а если вышел, то добавить к полученному коду 26 (число латинских букв), что обеспечит циклический сдвиг:

```
newcode := code - maxLen;
{ новый код }
{ цикличность }
if s[i] in ['a'..'z'] then
  if newcode < Ord('a') then
    newcode := newcode + 26;
if s[i] in ['A'..'Z'] then
  if newcode < Ord('A') then
    newcode := newcode + 26;
```

```
var c: char;
    s: string;
    len, maxLen, code, i, newcode : integer;
begin
  s := '';
  maxLen := 0;
  len := 0;
  { чтение данных }
  repeat
    read(c);
    s := s + c;
    if c in['a'..'z','A'..'Z'] then
      len := len + 1
    else
      begin
        if len > maxLen then
          maxLen := len;
        len := 0;
      end;
  until c = '.';
```

```
{ сдвиг кодов на maxLen влево }
 for i:=1 to Length(s) do
   if s[i] in ['a'..'z','A'..'Z'] then
     begin
       code := Ord(s[i]);  { старый код }
       newcode := code - maxLen; { новый код }
       { цикличность }
       if s[i] in ['a'..'z'] then
         if newcode < Ord('a') then
            newcode := newcode + 26;
        if s[i] in ['A'..'Z'] then
          if newcode < Ord('A')
             then newcode := newcode + 26;
       { запись нового кода }
       s[i] := Chr(newcode);
     end;
  writeln(s);
end.
```

Сайт К. Полякова
http://kpolyakov.narod.ru