Ларина Элла Семёновна
Учитель информатики высшей категории
МОУ «Лицей №2» г. Волгограда.

Тройка, семёрка, туз…, или ЕГЭ уж близится…

Задания, представленные на ЕГЭ по информатике, состоят на треть из задач по программированию. В заданиях школьнику нужно как самому написать программу, так и предсказать результат выполнения готовых программ. При помощи колоды карт можно смоделировать ячейки памяти, – выполняя трассировку программы, перекладывать карты с места на место, как бы меняя содержимое ячеек. Это позволит наглядно представить себе процесс пошагового выполнения программы, что особенно важно при большом наборе данных.

Представьте себе ситуацию – в школе на уроке ребята играют в карты. Не в укромном уголке, пряча колоду от посторонних глаз, а совершенно открыто, на глазах у учителя! Какова будет его реакция? Нарушителей дисциплины, скорее всего, призовут к порядку!

Разве что на уроке литературы креативный молодой педагог снисходительно отнесётся к желанию своих учеников глубже «погрузиться» в быт русских аристократов, изучив все тонкости любимого их времяпровождения (для лучшего понимания истинных причин аффектов, нервных срывов и умопомешательств при фразе «Сударь, Ваша карта бита!»).

А вот преподавателя точных наук совсем не волнует ни судьба Германа, ни несчастья Митеньки Карамазова! Хотя и на его занятии раскинуть пасьянс не помешало бы – почему бы не воспользоваться столь удобной моделью большого банка данных? Ведь так часто на уроке информатики идёт объяснение новой темы, основанное на ассоциациях (скажем, элементы массива – это пчелиные соты, или почтовые ящики в подъезде, или ... колода карт, разложенная в рядок на зеленом игральном сукне). С почтовыми ящи-

ками, а уж тем более с пчелиными сотами работа закончится уже на этапе ассоциаций – их ни потрогать, ни перевесить нельзя. Чего не скажешь о картах – принёс в школу колоду, разложил картишки на парте, и перекладывай с места на место. Сразу станет понятен и алгоритм сортировки одномерных массивов, и многие-многие другие!

Что ж, и мы воспользуемся этой удобной и наглядной моделью для изучения некоторых алгоритмов перебора элементов массива. Доставайте же колоду! Выкладывайте на парту «Тройку», «Семёрку», «Туза», ... ну, и «Пиковую даму» тоже.

Сейчас начнём, вот только немного вспомним материалы пройденных ранее на уроках тем. Готовы?

Типовые алгоритмы обработки одномерных массивов

Итак, повторяем. В разделе «Алгоритмизация» курса информатики средней школы изучают способы обработки данных. Данные хранятся в ячейках памяти (переменных), переменные имеют имена, различаются типами, содержимое переменных можно менять. А ещё переменные могут определённым образом группироваться друг с другом, образовывая массивы (одномерные массивы выстраиваются в ряд и имеют одно измерение – порядковый номер элемента массива в ряду; двумерные же массивы имеют два измерения, как, например, клетки шахматной доски).

Если мы храним данные в массиве, то нужно помнить, что все элементы массива имеют один и тот же тип (который нужно описывать при объявлении массива), одно и то же имя, но разные индексы (порядковые номера). Обращение к элементам массива происходит по имени (в нашем макете имя элемента Karta), индекс элемента пишут рядом с именем в скобках (Karta [1], Karta [2] и т.д.).

Karta [1]	Karta [2]	Karta [3]	Karta [4]
Тройка	Семерка	Туз	Дама

Как правило, обработка массива происходит путем циклического перебора всех его элементов по порядку. Каков номер шага цикла (хранящегося в счётчике цикла), таков и индекс элемента, к которому идет обращение на данном шаге. Выбранный элемент обрабатывается, затем наступает очередь следующего элемента и т.д.

Работа с массивами заключается в основном в одних и тех же стандартных заданиях (допустим, в нахождении суммы элементов массива, максимального элемента и т.д.), названных типовыми алгоритмами. Вот именно эти типовые алгоритмы обработки массивов и проходят в курсе средней школы при изучении программирования.

Вспомним эти типовые алгоритмы и, отталкиваясь от школьных знаний, попробуем расширить набор приёмов работы с массивами. Работу нижеприведённых алгоритмов проверьте, обращаясь к разложенным у вас на парте картам, имитирующим элементы массива в памяти компьютера.

Заполнение массива	Вывод массива
… readln (n); for i:=1 to n do readln (Karta[i]); …	. . . . . for i:=1 to n do writeln (Karta[i]); …
Сумма элементов	**Произведение элементов**

. . . . . .
s:=0;
for i:=1 to n do s:=s+Karta[i];
writeln (s);
…
	.....
	p:=1;
	for i:=1 to n do :=p*Karta[i];
	writeln (p);
	…

Поиск максимального элемента	**Поиск минимального элемента**

. . . . . .
max:=Karta[1];
for i:=1 to n do
if Karta[i]>max then
max:=Karta[i];
writeln (max);
…
	. . . . . .
	min:=Karta[1];
	for i:=1 to n do
	if Karta[i]<min then
	min:=Karta[i];
	writeln (min);
	…

В следующем блоке типовых алгоритмов предполагается, что обрабатываются только те элементы массива, которые соответствуют некоему заданному условию:

Сумма элементов, соответствующих условию отбора	Произведение элементов, соответствующих условию отбора

. . . . .
S:=0;
for i:=1 to n do
if {условие} then s:=s+Karta[i];
writeln (s);
…
	. . . . .
	p:=1;
	for i:=1 to n do
	if {условие} then
	p:=p*Karta[i];
	writeln (p);
	…

Количество элементов, соответствующих условию отбора	**Индексы элементов, соответствующих условию отбора**

. . . . . .
k:=0;
for i:=1 to n do
if {условие} then k:=k+1;
writeln (k);
…
	. . . . . .
	for i:=1 to n do
	if {условие} then writeln (i);
	…

Элемент, соответствующий условию отбора	
. . . . . . for i:=1 to n do If {условие} then Karta[i]:=…; …	

Выборка из массива по два элемента

Итого, типовые алгоритмы обработки одномерных массивов мы вспомнили. Во всех этих алгоритмах осуществлялся пошаговый выбор одного элемента из массива.

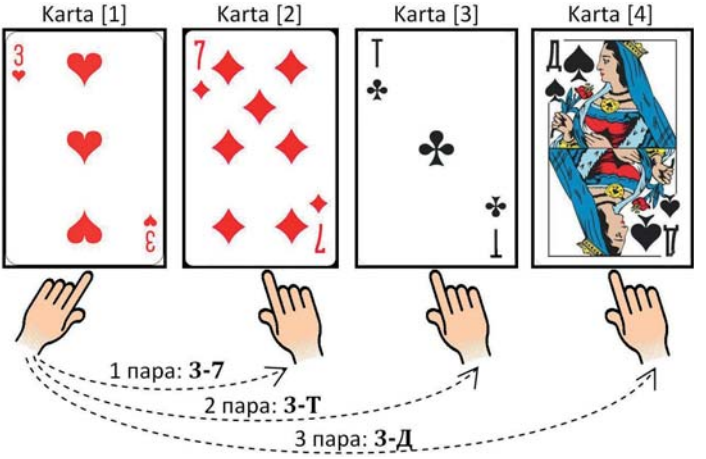
А что если **выбирать нужно ПО ДВА элемента (пусть это будут рядом расположенные элементы)**? И, предположим, с выбранными элементами необходимо произвести те же действия (что и в выше рассмотренных алгоритмах) – допустим, найти пару рядом расположенных элементов целочисленного массива, сумма которых максимальна и чётна (диапазон возможных значений элементов – от -10 до 10).

Программная реализация задачи такова:

```
. . .
s:=0;
max:=-101;
for i:=1 to n-1 do
    begin
    s:=Karta[i]+Karta[i+1];
    if  (s>max) and (s mod 2=0) then
                      begin
                      max:=s; b:=i
                      end;
    end;
if max=-101 then writeln ('не найдено')
          else writeln bi, b+1);
. . .
```

Пойдём дальше. А что если нужно **выбирать ПО ДВА, но НЕ ОБЯЗАТЕЛЬНО РЯДОМ расположенных элемента**?

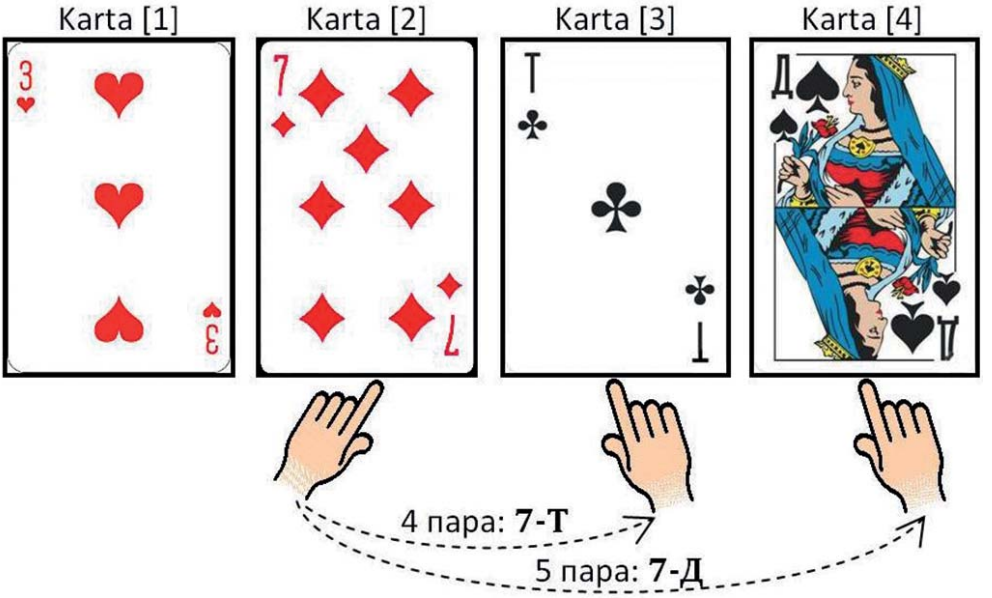
Сразу отметаем случайную выборку двух элементов из массива – запутаемся! Определимся с порядком перебора элементов, для чего опять кладём перед собой 4 карты из колоды. Указательный палец левой руки ставим на первую карту («Тройку»), правая же рука поочерёдно указывает на вторую, затем третью, затем четвёртую карту, которые будут по очереди выбраны в паре с первой. Итого, выбраны пары карт: «3–7», «3–Туз», «3–Дама».



Фрагмент алгоритма выборки пары карт приведен на языке программирования Паскаль:

```
. . . . . .
for j:=2 to 4 do
    begin
    {пара: karta[1] и karta[j]}
    end
. . . . . .
```

Далее левая рука переходит ко второй карте и никуда с неё не двигается. Правая же рука в процессе перебора будет поочерёдно выбирать пару для первой выбранной карты (для карты Karta[2]). Вот эти пары: «7–Туз», «7–Дама».

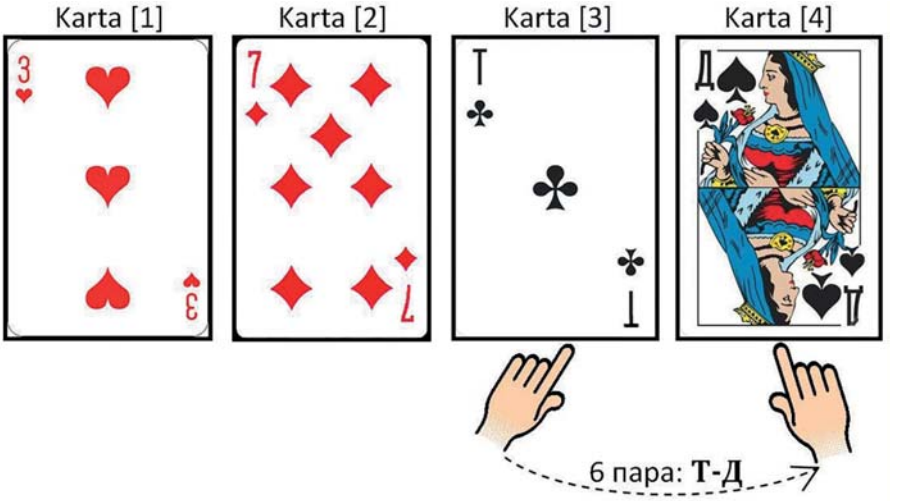


Обратите внимание, что мы не выбрали пару «7–3», так как она уже была выбрана раньше (порядок пар в паре был такой «3–7»). Разные принципы выборки элементов рассматриваются в разделе «Комбинаторика» и подразделяются на размещения, сочетания и перестановки. Не вдаваясь в подробности, уточним только, что в зависимости от задачи могут считаться разными пары, в которых позиция элементов имеет значение либо не имеет. Будем считать, что в нашем случае пары «7–3» и «3–7» идентичны.

Программная реализация формирования этих пар:

```
. . . . . .
for j:=3 to 4 do
    begin
    {пара: karta[2] и karta[j]}
    end
. . . . . .
```

Следующим шагом будет перемещение левой руки на третью карту. Правая рука должна опять перебрать все карты от четвёртой (она следует за третьей) и до последней карты в наборе (в нашем случае на четвёртой же и остановимся).



Фрагмент программного кода:

```
. . . . . .
for j:=4 to 4 do
    begin
    {пара: karta[3] и karta[j]}
    end
. . . . . .
```

Все приведённые выше алгоритмы выборки двух карт практически одинаковы, за исключением индексов-номеров первой карты из пары и начального значения счётчика цикла (на единицу большего, чем номер первой карты в паре). Повторяющиеся фрагменты оформим в цикле, а индекс первой карты пары привяжем к счётчику внешнего цикла. В итоге получим алгоритм, при помощи которого мы осуществили выборку всех пар карт из набора, лежащего перед нами на столе (Karta[i] – это карта-элемент в левой руке, Karta[j] – карта-элемент в руке правой):

```
. . . . . .
for i:=1 to n-1 do
for j:=i+1 to n do
begin
   {пара: Karta[i] – Karta[j]}
end
. . . . . .
```

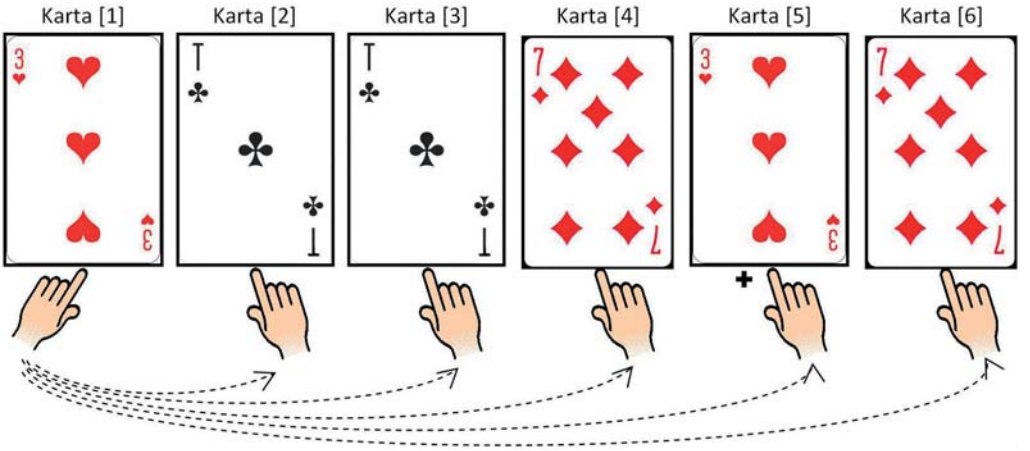
Использование «флажков» для фиксации некоторого события

Идём далее. А что же конкретно можно делать с выбранными парами карт?

Предположим, что **в школе два ученика смешали свои карты в одну колоду. Как опять разделить весь набор на две колоды?**

Все просто, нужно выложить в ряд все карты и опять организовать перебор, рассмотренный выше. Карту, на которую указывает левая рука, сравниваем с картой, на которую указывает правая рука. При их совпадении рисуем мелом знак (крестик) на сукне под картой в правой руке. Когда будут перебраны все пары карт, мы увидим под выложенной колодой меловые крестики – они указывают на карты одной колоды. Те же карты, под которыми не окажется пометок, принадлежат другой колоде.

Первый «проход» – сравниваем первую карту со второй, затем с третьей и т.д. Совпадение будет с пятой картой. Под ней и ставим крестик.



Далее по алгоритму, рассмотренному ранее.

Но как же в программном коде реализовать рисование крестика под выбранной картой? Для программной реализации будем использовать дополнительный массив так называемых флажков. В него будем заносить «метки» при выборе элемента (можно и крестики-плюсики, можно единички – при определении совпадения элементов):

Karta [1]	Karta [2]	Karta [3]	Karta [4]	Karta [5]	Karta [6]
Тройка	Туз	Туз	Семерка	Тройка	Семерка

Flag [1]	Flag [2]	Flag [3]	Flag [4]	Flag [5]	Flag [6]
0	0	1	0	1	1

```
. . . . . .
for i:=1 to n-1 do
        for j:=i+1 to n do
                  if Karta[i]=karta[j] then flag[j]:=1;
for i:=1 to n do
         if flag[i]=0 then write (Karta[i]);
. . . . . .
```

В массиве флажков можно фиксировать любое событие: можно вести подсчёт количества встречаемости элемента, «вычёркивать» элемент (занося во «флажок», предположим, «-1») и т.д.

Допустим, что несколько карт из колоды потеряны (колода неполная). В таком случае можно найти:

- какая карта встречается в колоде единожды;
- каких карт в колоде больше;
- сколько раз встречается каждая карта в колоде;
- отсортировать карты по частоте их встречаемости.

Решение всех этих задач сводится к одному – при совпадении карт в выбранной паре нужно flag[i] (а как мы уже знаем, счётчик внешнего цикла i указывает на положение левой руки) нарастить на 1, а в flag[j] (счётчик внутреннего цикла j управляет правой рукой) поместить «-1» (тем самым как бы вычеркнуть повторяющуюся карту-элемент, т.к. совпадение с ней уже зафиксировано и дальнейшая её проверка на совпадение с другими картами должна быть исключена):

```
var      Karta: array [1..5] of string;
flag: array [1..5] of integer;
i,j,n: integer;
begin
Readln (n);
for i:=1 to n do readln (Karta[i]);
writeln;
for i:=1 to n-1 do
      if flag[i]<>-1 then
                 begin
                 for j:=i+1 to n do
                      if Karta[i]=karta[j] then
                                    begin
                                    flag[i]:=flag[i]+1;
                                    flag[j]:=-1;
                                    end;
                  end;
for i:=1 to n do
      if flag[i]=0 then write (Karta[i],' ');
end.
```

В приведённой программе в дополнительном массиве флажков накапливается количество повторений каждой карты, поэтому для реализации второй задачи (нахождение карты, которая встречается в колоде чаще других) нужно лишь найти максимальный элемент мас-

сива flag и по его индексу вывести соответствующий (имеющий такой же индекс) элемент массива Karta.

Для сортировки карт по частоте встречаемости применяем алгоритм сортировки для массива флажков, не забывая при этом менять местами также и соответствующие элементам массива flag элементы массива Karta. При выводе не выводим карты, флажки-указатели на которые равны «-1».

Различные варианты выборки второго элемента из пары элементов массива

А теперь задачка из банка данных ЕГЭ. Есть несколько карт. Можно ли из них составить палиндром (палиндром одинаково читается как слева направо, так и справа налево)?

Подумайте перед тем, как прочитать решение данной задачи!

Подумали? Наверное, и вы решили подсчитать количество повторений каждой карты в наборе. Палиндром можно получить, если каждая карта встречается четное число раз (половина будет располагаться слева от середины, половина – справа), и лишь ОДНА карта («центральная») может быть представлена в нечётном количестве (но таковых может и не быть вовсе).

Попытайтесь самостоятельно доработать этот алгоритм и вывести карты, которые будут образовывать палиндром (при возможности), т.е. составить палиндром из предложенного набора карт.

А теперь вопрос: а **можно ли иначе организовать выборку двух карт из массива**? В рассмотренном выше приёме мы фиксировали левой рукой карту, а правая рука поэтапно перемещалась от следующей за зафиксированной карты **до конца колоды**. Но ведь правая рука может двигаться и **от начала колоды** до зафиксированной левой рукой карты:

```
. . . . . . . . . .
for i:=2 to n do
for j:=1 to i-1 do
{Пара: Karta[i] - Karta[j]}
. . . . . . . . . .
```

Не принципиально, и в этом случае будут выбраны все пары карт из колоды. Но для решения некоторых задач метод обхода имеет значение.

Задача «Камера хранения». Имеется запись прихода и ухода посетителей камеры хранения на вокзале (запись упорядочена по времени прихода посетителей). Какие номерки получат посетители, если перед приёмщиком багажа стоит задача – количество выдаваемых номерков свести к минимуму?

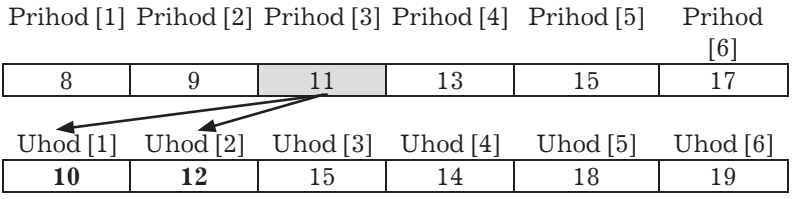
Хранить время прихода, как и время ухода, будем в массивах Prihod и Uhod (элементы массивов с одним и тем же индексом соответствуют приходу-уходу одного и того же посетителя). Для хранения жетона с номером, выданным посетителю, также будем использовать массив – Nomer. В переменной Novij будем готовить новый номерок, на случай, если все номерки находятся на руках у посетителей и никто ещё номерок не вернул.

Prihod [1]	Prihod [2]	Prihod [3]	Prihod [4]	Prihod [5]	Prihod [6]
8	9	11	13	15	17

Uhod [1]	Uhod [2]	Uhod [3]	Uhod [4]	Uhod [5]	Uhod [6]
10	12	15	14	18	19

Nomer [1]	Nomer [2]	Nomer [3]	Nomer [4]	Nomer [5]	Nomer [6]
1	2	1	2	2	1

Для решения задачи будем обходить массив Prihod, сравнивая значение очередного его элемента со всеми элементами массива Uhod (начиная с первого и до элемента, анализируемого на данном этапе). В общем-то, уже понятно – правая рука фиксирует элемент (карту, если мы работаем с моделью «Карточная колода»), а левая рука поочерёдно проходит все элементы от первого до того, что зафиксирован правой рукой (такой алгоритм обхода мы уже рассматривали выше, но теперь внесём небольшое отступление – правая рука «путешествует» по массиву Prihod, левая же – по массиву Uhod).



Если время прихода очередного посетителя больше времени ухода кого-то из предыдущих (и при этом его номерок не отдан другому, что будем фиксировать в массиве Nomer пометкой «-1»), то его номерок передаем вновь прибывшему.

Если посетитель пришел, а все номерки «на руках», то ему выдается резервный номерок (Novij). Сейчас же создается новый резервный номерок (значение Novij увеличивается на 1).

```
const n=6;
var Prihod, Uhod, Nomer: array [1..n] of integer;
      i, j, Novij: integer;
begin
for i:=1 to n do readln (Prihod[i], Uhod[i]);
Nomer[1]:=1; Novij:=2;
write (Nomer[1],' ');

for i:=2 to n do
      begin
      for j:=1 to i-1 do
            if (Prihod[i]>Uhod[j]) and (Nomer[j]<>-1) then
                  begin
                  Nomer[i]:=Nomer[j];
```

```
                    Nomer[j]:=-1;
                    end;
        if Nomer[i]=0 then
                    begin
                    Nomer[i]:=Novij;
                    Novij:=Novij+1;
                    End;
    write (Nomer[i],' ');
        end;
end.
```

Алгоритм предполагает, что если сдано более одного номерка (несколько посетителей к моменту прихода очередного уже успели забрать свой багаж и вернуть служащему номерок), то вновь прибывшему будет вручен номерок последнего ушедшего посетителя камеры хранения.

Итак, мы освоили алгоритм перебора элементов массива по парам и использовали его в решении различных задач.

Не прячьте далеко колоду карт – ЕГЭ уж близится, а метод отрабатывать нужно! Откройте демоверсию ЕГЭ по информатике и попытайтесь решить некоторые задачи по программированию, вооружившись игральными картами, вот увидите – Тройка, Семёрка и Туз вас не подведут!

Юмор Юмор Юмор Юмор Юмор Юмор

Вполне серьёзный ответ

Трёхлетний внук выдающегося физика-теоретика Владимира Александровича Фока играл на даче в саду. Услышав, что растения, среди которых он бегает, называются папоротниками, спросил:

– Дедушка, а маморотники есть?

– Нет

– Почему?

– Но зато есть мамонты и нет папонтов.

Редкая книга

Ни одна книга не вызывает после своего выхода столько разговоров, сколько телефонная.

Какое вы хотите приобрести право?

Въезжающего в некий город N автомобилиста встречает такое объявление: «Если вы едете со скоростью 40 км/ч, вам предоставляется право посетить нашу образцовую тюрьму. Скорость свыше 60 км/ч даёт вам право познакомиться с нашей превосходной больницей. А если вы превысите 80 км/ч, вас ждёт наше показательное кладбище.