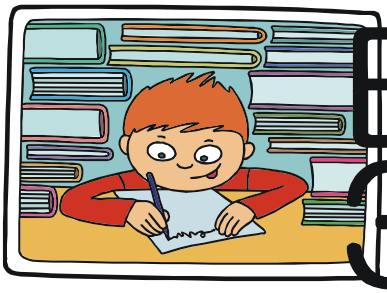


Впереди Экзамены

Богомолова Ольга Борисовна,
Усенков Дмитрий Юрьевич

ПРОГРАММИРОВАНИЕ: НЕДЕСЯТИЧНЫЕ СИСТЕМЫ СЧИСЛЕНИЯ

Не секрет, что программирование составляет основу школьного курса информатики, и потому на экзамене этой теме уделяется значительное внимание. Программированию и алгоритмике посвящен целый ряд задач в первой части заданий ЕГЭ, а во второй части три из четырех задач – именно по программированию.

Одна из таких задач – та, что под номером 25 (как раз во второй части ЕГЭ), – раньше не вызывала у учащихся больших трудностей: алгоритмы подсчета суммы и произведения элементов массива, определения количества элементов, удовлетворяющих некоторым условиям, или поиска минимума/максимума всегда изучаются в курсе информатики и большинством учащихся неплохо усваиваются даже классе в седьмом. Но, увы, разработчики заданий ЕГЭ, похоже, «спят и видят», как бы усложнить жизнь и учителям и учащимся, и стараются напридумывать такие задания, для которых *в существующих учебниках даже нет соответствующих объяснений*!

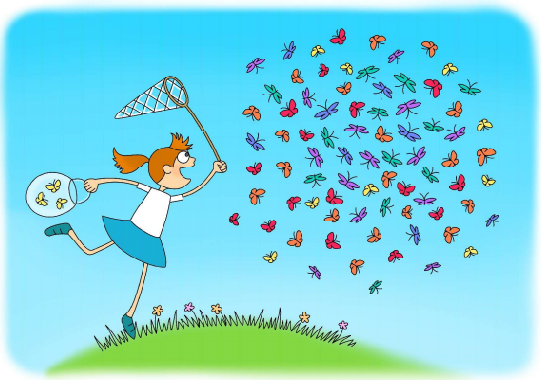
Не стала исключением из этого печального правила и задача 25. На очередном тренаже с ней из 10 и 11 классов справились лишь несколько наиболее сильных учеников. Впрочем, задачи такого типа вполне решаемы, если вспомнить теорию из темы «системы счисления».

Начнем с разбора задачи, аналогичной прежним заданиям 25.

Задача 1. Дан массив, содержащий 3000 положительных целых чисел, не превышающих 10000. Необходимо найти и вывести сумму таких элементов этого массива, запись которых содержит ровно три цифры, и при этом первая цифра не равна последней.

Например, для массива из 6 элементов, содержащего числа 353, 245, 37, 2345, 123, 88, ответ будет равен 368: числа 37, 88 и 2345 не подходят, так как в них не три цифры, а число 353 хотя и трехзначное, но его первая и последняя цифры равны.

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



```
Const
N=3000;
var a: array [1..N] of integer;
       i, s, k: integer;
begin
  for i:=1 to N do readln(a[i]);
  . . .
end.
```

Решение

Проанализируем условие задачи.

Указано, что количество чисел в массиве равно 3000, что все числа положительные и целые и что эти числа не превышают 10000. Сразу отметим: эти данные требуются *только для правильного объявления массива и переменных* – количество (3000) определяет размер массива (и он уже задан в условии – **`N=3000`**), а верхний предел значений чисел 10000 указывает, что тип массива и соответствующих переменных должен быть **`integer`**.

Заметим также, что в приведенном в условии фрагменте текста программы уже есть не только *правильное* объявление массива и рабочих переменных, но и цикл чтения значений элементов массива, так что нам нужно написать только «смысловую» часть программы для их обработки. И обработка эта – *типовой алгоритм вычисления суммы элементов массива, удовлетворяющих заданному условию*. То есть, заготавливается (изначально обнуляется) переменная-счетчик, к которой будут прибавляться элементы массива, а проверка соответствия их значений заданному условию осуществляется в операторе **`if`** (соответственно, прибавление элемента к переменной-счетчику производится в ветви **`then`**).

1. Как проверить, что число (элемент массива) содержит ровно три цифры? Очевидно, надо сравнить этот элемент с граничными значениями: он должен быть больше, чем самое большое двузначное число (99), и меньше, чем самое маленькое четырехзначное (1000). То есть, соответствующую часть условия можно записать так:

```
(A[i] > 99) and (A[i] < 1000)
```

Возможен, впрочем, и альтернативный вариант: элемент массива должен быть больше или равен самому маленькому трехзначному числу (100) и меньше или равен самому большому трехзначному (999). Тогда условие выглядит так:

```
(A[i] >= 100) and (A[i] <= 999)
```

Можно, конечно же, и комбинировать эти варианты, например:

```
(A[i] >= 100) and (A[i] < 1000)
```

или

```
(A[i] > 99) and (A[i] <= 999)
```

Будьте внимательны! Очень частая ошибка учащихся состоит в том, что они проверяют только часть этого условия – что количество цифр в числе не менее заданного либо не более заданного, то есть вместо двух указанных выше сравнений записывают только какое-то одно. **Не забывайте сравнивать число с обеими границами нужного интервала значений!**

2. Как проверить, что первая и последняя цифры не равны друг другу? Для этого надо сначала выделить из числа эти цифры. Как это сделать?

Последнюю цифру можно выделить как остаток от деления исходного числа на 10, воспользовавшись операцией **`mod`** (в языке Паскаль; в других языках тоже есть подобная операция, но, возможно, записываемая иначе).

Первую же цифру можно выделить при помощи операции целочисленного деления исходного числа на 100 (с отбрасыванием остатка). Очевидно, что с учетом уже проверенного условия «число имеет ровно три цифры» операция **`A[i] div 100`** как раз дает нам требуемую первую цифру.

Итого получаем следующее условие проверки неравенства первой и последней цифры:

```
(A[i] mod 10) <> (A[i] div 100)
```

Ну, а теперь нетрудно написать и всю требуемую «смысловую» часть программы:

```
  s := 0;  // обнулили счетчик суммы
   for i:=1 to N do
     if (A[i] >= 100) and (A[i] <=
999) and ((A[i] mod 10) <> (A[i]
div 100)) then s := s + A[i];
```

Остается только добавить вывод результата:

```
writeln(s);
```

И задача решена. Полный текст программы будет таким:

```
Const
N=3000;
var a: array [1..N] of integer;
       i, s, k: integer;
begin
  for i:=1 to N do readln(a[i]);
  s := 0;  // обнулили счетчик суммы
  for i:=1 to N do
    if (A[i] >= 100) and (A[i] <=
999) and ((A[i] mod 10) <> (A[i]
div 100)) then s := s + A[i];
  writeln(s);
end.
```

А теперь познакомимся с новой задачей, которую придумали «добрые дяди и тети» – разработчики заданий ЕГЭ (это, конечно, не в точности та самая задача из тренажа, о которой говорилось вначале, но аналогичная ей).

Задача 2. Дан массив, содержащий 5000 натуральных чисел, не превышающих 10000. Требуется найти и вывести количество элементов этого массива, тринадцатеричная запись которых содержит ровно четыре знака и первая цифра больше третьей.

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

```
Const
N=5000;
var a: array [1..N] of integer;
       i, s, k: integer;
begin
  for i:=1 to N do readln(a[i]);
  . . .
end.
```

На первый взгляд, решение должно быть очень сложным – нужно сначала перевести исходное *десятичное* число, хранящееся в массиве, в тринадцатеричную (!) систему счисления, потом проверить, сколько в этой записи получится цифр (операция вычисления длины символьной строки?), потом выделить требуемые цифры и сравнить их как числа.

Но на самом деле всё гораздо проще: эта «грозная» задача решается полностью аналогично предыдущей, где условие сформулировано для десятичной системы счисления.

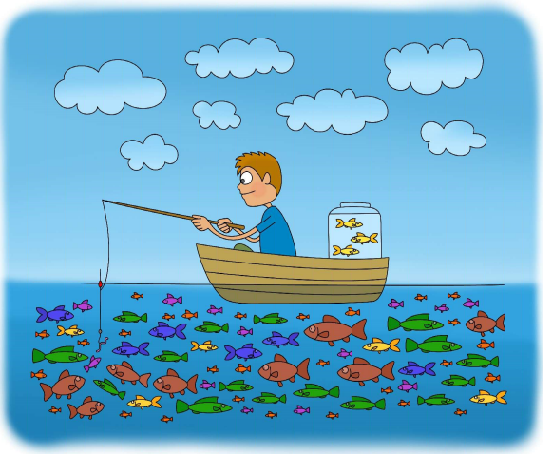
Решение

Сразу займемся смысловой частью программы.

1. Как проверить, что исходное десятичное число в *тринадцатеричной* системе счисления имеет ровно 4 знака?

Аналогично предыдущей задаче, надо сравнить наше число (временно «забыв», что оно десятичное) с наименьшим и наибольшим возможными тринадцатеричными значениями, имеющими ровно 4 цифры. Очевидно, что это числа 1000_{13} и $CCCC_{13}$. (Напомним: в системах счисления с основанием больше 10 используются «цифры» A = 10, B = 11, C = 12, D = 13, E = 14, F = 15 и т. д., а в тринадцатеричной системе счисления значения цифр ограничены 12-ю.)

Теперь вспомним, что исходное число у нас все же десятичное. Значит, найденные «граничные» тринадцатеричные значения надо пересчитать в десятичные:



$1000_{13} = 13^3 = 2197_{10}$,
$CCCC_{13} = 10000_{13} - 1 = 13^4 - 1 =$
$= 28561 - 1 = 28560$.

И теперь записываем соответствующую часть условия по аналогии с тем, как это делалось в предыдущей задаче:

```
(A[i] >= 2197) and (A[i] <= 28560)
```

Либо (тоже по аналогии с предыдущей задачей) можно использовать «граничные» значения вне допустимого интервала, но со строгими неравенствами:

```
(A[i] > 2196) and (A[i] < 28561)
```

или же скомбинировать эти два варианта записи.

2. А как выделить из исходного *десятичного* числа первую и третью цифры *тринадцатеричной* записи? Тоже по аналогии с прежней, «десятичной» задачей, используя операцию вычисления остатка от деления и операцию целочисленного деления исходного числа – только уже на основание заданной системы счисления в соответствующей степени.

Выделяем первую («старшую») цифру числа – для этого нужно целочисленное деление на 13 в степени 3 (то есть целочисленное деление на число 1000_{13}); это значение мы, кстати, уже вычисляли:

```
A[i] div 2197
```

А чтобы выделить третью цифру (ту, что во второй позиции справа), нам нужно сначала «отсечь» часть числа, где эта цифра будет «старшей», то есть вычислить остаток от деления исходного числа на 100_{13}, а потом выделить эту оставшуюся «старшей» цифру операцией целочисленного деления на 10_{13}. Только, конечно, эти тринадцатеричные значения надо сначала пересчитать в десятичные:

$100_{13} = 13^2 = 169_{10}$,
$10_{13} = 13_{10}$.

Тогда операция выделения из исходного числа третьей цифры имеет вид:

```
(A[i] mod 169) div 13
```

Ну, а соответствующая часть условия, проверяющая, что первая цифра в тринадцатеричной записи больше третьей, будет такой:

```
(A[i] div 2197) > ((A[i] mod 169) div 13)
```

Проще всего для такого решения сначала записать «типовые» операции с десятичными числами, потом вспомнить, что каждое десятичное константное значение – это на самом деле число в соответствующей системе счисления, а затем пересчитать получившиеся «недесятичные» константы в десятичные числа. Например, для этой задачи последовательность действий будет такая:

Первая цифра 4-значного числа:		***Третья цифра 4-значного числа:***
`A[i] div 1000`	← *было бы для десятичного значения:* →	`(A[i] mod 100) div 10`
`A[i] div 1000`$_{13}$	← *для 13-ричной системы счисления* →	`(A[i] mod 100`$_{13}$`) div 10`$_{13}$
`A[i] div 2197`	← *с пересчетом в десятичные значения* →	`(A[i] mod 169) div 13`

Задача практически решена. Остается правильно записать весь оператор **if**, проверяющий заданные условия, а в его ветви **then** записать вычисление количества.

```
  k := 0;  // обнуляем переменную-
           // счетчик
  for i:=1 to N do
    if (A[i] >= 2197) and (A[i] <=
28560) and (A[i] div 2197) > ((A[i]
mod 169) div 13) then k := k + 1;
  writeln(k);  // вывод результата
```

(Отметим, кстати, что реальная задача в тренаже была гораздо проще, чем разобранная здесь: там речь шла о более «привычной» шестнадцатеричной записи, а сравнивать требовалось первую и последнюю цифры двузначного шестнадцатеричного числа.)

И «на закуску», для закрепления изученного материала, рассмотрим еще одну подобную задачу.

Задача 3. Дан массив, содержащий 4000 натуральных чисел, не превышающих 10000. Требуется найти и вывести произведение элементов этого массива, восьмеричная запись которых содержит ровно два знака и сумма цифр этой записи больше десятичного значения 10.

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

```
Const
N=4000;
var a: array [1..N] of integer;
       i, p, k: integer;
begin
  for i:=1 to N do readln(a[i]);
  . . .
end.
```

Решение

1. Проверяем, что в восьмеричной системе число имеет ровно два знака.

Минимальное двузначное восьмеричное число равно $10_8 = 8_{10}$.

Максимальное двузначное восьмеричное число равно $77_8 = 100_8 - 1 = 8^2 - 1 = 64 - 1 = 63$.

Фрагмент условия:

```
(A[i] >= 8) and (A[i] <= 63)
```

или

```
(A[i] > 7) and (A[i] < 64)
```

2. Выделяем цифры восьмеричной двузначной записи:

– первая цифра – целочисленным делением на 10_8: **A[i] div 8**;

– последняя цифра – вычислением остатка от деления на 10_8: **A[i] mod 8**.

Тогда соответствующая часть условия:

```
(A[i] div 8) + (A[i] mod 8) > 10
```

Запись «смыслового» фрагмента программы:

```
  p := 1;
// переменная для вычисления
// произведения изначально
// приравнивается единице
  for i:=1 to N do
    if (A[i] >= 8) and (A[i] <= 63)
and ((A[i] div 8) + (A[i] mod 8) >
10) then p := p * A[i];
  writeln(p);  // вывод результата
```

Итак, остается сделать следующие выводы:

1. Бояться подобных задачек с недесятичными системами счисления не нужно. Решаются они точно так же, как и в десятичной системе счисления.

2. Для проверки, что исходный элемент массива имеет ровно заданное количество цифр, нужно сравнить это число с наименьшим значением, которое имеет в заданной системе счисления заданное количество цифр, и с наибольшим значением, которое

имеет в заданной системе счисления заданное количество цифр. Только полученные «граничные» значения в заданной систем счисления надо не забыть пересчитать в соответствующие им десятичные.

3. Выделение цифр из записи исходного числа в заданной системе счисления производится при помощи операций целочисленного деления (**div**) и вычисления остатка от деления (**mod**) на числа, равные «десяткам», «сотням», «тысячам» и т. д. в соответствующей системе счисления:

– выделить последнюю цифру можно при помощи операции **mod**,

– выделить первую цифру можно при помощи операции **div**,

– выделить цифру из середины числа можно сочетанием операций **mod** и **div**.

Богомолова Ольга Борисовна,
доктор педагогических наук,
почетный работник сферы образования Российской Федерации,
учитель информатики и математики ГБОУ СОШ № 1360, г. Москва,

Усенков Дмитрий Юрьевич,
Московский государственный институт индустрии туризма имени Ю.А. Сенкевича, г. Москва.