C4 (высокий уровень, время – 60 мин)

Тема: Обработка данных, вводимых в виде символьных строк (написать программу средней сложности из 30-50 строк).

Что нужно знать:

- символьная строка – это цепочка символов, которая может обрабатываться как единое целое
- для обращения к символу с номером **i** строки **s** используется запись **s[i]**, это говорит о том, что строка – особый вариант массива, в котором хранятся символы
- знак сложения при работе с символьными строками означает сцепку, объединение двух строк в одну (добавление второй строки в конец первой), например:
  ```
  s := '123' + '456'; { получили '123456' }
  ```
- с помощью функции **Ord** можно получить код символа; цифры имеют коды от 48 (цифра 0) до 57 (цифра 9), например
  ```
  k := Ord('1');    { получили 49 }
  ```
  то же самое можно сделать с помощью преобразования типа (привести **char** к **integer**)
  ```
  k := integer('1'); { получили 49 }
  ```
- с помощью функции **Chr** можно сделать обратный переход: получить символ по его коду, например
  ```
  c := Chr(49);  { получили символ '1' }
  ```
  то же самое можно сделать с помощью преобразования типа (привести **integer** к **char**)
  ```
  c := char(49); { получили символ '1' }
  ```
- для работы со строками в наиболее распространенных Паскаль-средах (*Turbo Pascal, Borland Pascal, PascalABC,* среда *АЛГО*) используют стандартные функции (здесь **s** – это переменная типа **string**, символьная строка; **n** и **r** – целые переменные)

`n := Length(s);`	записать длину строки **s** в целую переменную **n**
`s1 := Copy(s, 2, 5);`	записать в символьную строку **s1** подстроку строки **s**, которая начинается с символа с номером 2 и состоит из 5 символов (важно – не со 2-го по 5-ый символ!)
`n := Pos('Вася', s);`	записать в целую переменную **n** номер символа, с которого в строке **s** начинается подстрока 'Вася' (если ее нет, в переменную **n** записывается 0); так же можно искать отдельные символы (важно: сначала указываем, **что** ищем, а потом – **где**)
`n := StrToInt(s);`	преобразовать строку **s** в целое число и записать результат в переменную **n** (*PascalABC, Delphi*)

  и процедуры

`Delete(s, 2, 5);`	удалить из строки **s** 5 символов, начиная со второго
`Insert('Вася', s, 3);`	вставить в строку **s** фрагмент 'Вася', начиная с третьего символа (между 3-м и 4-м)
`Val(s, n, r);`	преобразовать строку **s** в целое число и записать результат в переменную **n**; если при этом произошла ошибка, в переменной **r** будет номер ошибочного символа, если все нормально – ноль

- структура (в Паскале она называется «запись», *record*) – это сложный тип данных, который может включать в себя несколько элементов – полей; поля могут иметь различный тип
- записи в Паскале объявляются с помощью ключевого слова **record**; в простейшем случае можно выделить память под одну запись так:

```
var x: record
     name: string;
     code: integer;
    end;
```

эта запись состоит из двух полей: символьной строки **name** и целого числа **code**

- записи очень удобны для работы, когда все данные в целом представляют собой единый блок информации, например, данные об ученике; если не использовать записи, было бы нужно выделять в памяти отдельно символьную строку и отдельно целую переменную, причем эти данные внешне были бы никак не связаны, поэтому программа с записями часто получается логичнее и понятнее как для автора, так и для того, кто будет в ней разбираться
- для обращения к полям записи используют точку, например **x.name** означает «поле **name** записи **x**»
- можно сразу объявить массив записей:

```
var Info: array[1..100] of record
     name: string;
     code: integer;
    end;
```

это 100 одинаковых записей, имеющих общее имя **Info** и расположенных в памяти рядом; в каждой структуре есть поля **name** и **code**; чтобы работать с полями записи с номером **k** используют обращения вида **Info[k].name** и **Info[k].code**

Сложность алгоритмов:

- обозначение $O(N)$ говорит о том, что при увеличении в 2 раза размера массива данных количество операций тоже увеличивается примерно в 2 раза (для больших N)
- сложность $O(N)$ имеет алгоритм с одним или несколькими простыми (не вложенными!) циклами в каждом из которых выполняется N шагов (как при поиске минимального элемента)
- количество операций для алгоритма, имеющего сложность $O(N)$, вычисляется по формуле $p = a \cdot N + b$, где a и b – некоторые постоянные
- если в одном алгоритме решения задачи используется несколько циклов от 1 до N, а во втором – только один цикл, то алгоритм с одним циклом, как правило, эффективнее (хотя оба алгоритма имеют сложность $O(N)$, постоянная a в каждом случае своя, для алгоритма с несколькими циклами она будет больше)
- для алгоритма, имеющего сложность $O(N^2)$, количество операций пропорционально квадрату размера массива, то есть, если N увеличить в 2 раза, то количество операций увеличивается примерно в 4 раза (например, в программе используется два вложенных цикла, в каждом из которых N шагов); сложность $O(N^2)$ имеют простые способы сортировки массивов: метод «пузырька», метод выбора
- при больших N функция $f_1(N) = a_1 N^2$ растет значительно быстрее, чем $f_2(N) = a_2 N$, поэтому алгоритм, имеющий сложность $O(N^2)$ всегда менее эффективен, чем алгоритм сложности $O(N)$
- иногда встречаются алгоритмы сложности $O(N^3)$ (три вложенных цикла от 1 до N), при больших N они работают медленнее, чем любой алгоритм сложности $O(N^2)$, то есть, менее эффективны
- для многих задач известны только алгоритмы экспоненциальной сложности, когда размер массива входит в показатель степени, например $O(2^N)$, для больших N такие задачи не решаются за приемлемое время (например, «взламывание» шифров)

Пример задания:

На вход программе подаются сведения о номерах школ учащихся, участвовавших в олимпиаде. В первой строке сообщается количество учащихся N, каждая из следующих N строк имеет формат:

<Фамилия> <Инициалы> <номер школы>

где <Фамилия> – строка, состоящая не более чем из 20 символов, <Инициалы> – строка, состоящая из 4-х символов (буква, точка, буква, точка), <номер школы> – не более чем двузначный номер. <Фамилия> и <Инициалы>, а также <Инициалы> и <номер школы> разделены одним пробелом. Пример входной строки:

Иванов П.С. 57

Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет выводить на экран информацию, из какой школы было меньше всего участников (таких школ может быть несколько). При этом необходимо вывести информацию только по школам, пославшим хотя бы одного участника. Следует учитывать, что N>=1000.

Как правильно понимать условие?

1) на первый вопрос – как именно вводятся данные – находим ответ в самом начале условия: вроде бы «дежурная» фраза «на вход программе подаются…» означает, что данные нужно читать не из файла, а со стандартного входного потока; это, в свою очередь, значит, что можно использовать привычные операторы **read**(**readln**), предполагая, что кто-то вводит эти данные с клавиатуры вручную[1]
2) итак, сначала вводится количество записей в файле **N**, а затем **N** строк с информацией; заметим, что из всей этой информации нас интересует (в каждой строке) только номер школы, остальное можно просто отбрасывать
3) номер школы стоит после второго пробела в строке
4) «*<номер школы> – не более чем двузначный номер*» – крайне важная информация; собственно, только она и позволяет найти хорошее решение задачи; это значит, что школ не более 99!
5) что означает выражение «как можно более эффективная программа»?
   - прежде всего, данные читаются только один раз, за один проход, нельзя «вернуться» и прочитать что-то вновь
   - в программе не выполняются никакие лишние действия
   - используемые алгоритмы имеют минимальную сложность (см. выше)
   - расходуется минимальный возможный объем памяти; например, чтобы найти количество отрицательных элементов массива, не нужно вводить второй массив; если нам достаточно держать в памяти одну введенную строку, не нужно одновременно хранить все прочитанные строки
6) зачем нужно уточнение «*N>=1000*»? этим авторы задачи намекают на то, что не нужно считывать все данные в оперативную память, а потом уже их обрабатывать; основная обработка должна быть сделана сразу, в том же цикле, где читаются входные данные
7) мы будем считать, что в исходных данных нет ошибок (так принято на олимпиадах и экзаменах), иначе обработка разнообразных ошибок будет составлять основную часть программы

[1] Или используется перенаправление входного потока из командной строки, но это уже абсолютно неважно…

Решение:

1) по условию, единственная информация, которая нам нужна в итоге для вывода результата – это количество участников по каждой школе
2) так как номер школы состоит (по условию!) не более, чем из двух цифр, всего может быть не более 99 школ (с номерами от 1 до 99)
3) поэтому можно ввести массив **C** из 99 элементов; для всех **k** от 1 до 99 элемент **C[k]** будет ячейкой-счетчиком, в которой накапливается число участников от школы с номером **k**; сначала во все элементы этого массива записываются нуль (обнуление счетчиков):
```
for k:=1 to 99 do C[k]:=0;
```
во многих системах программирования на Паскале все глобальные переменные автоматически обнуляются, и таким образом, этот цикл ничего не дает; однако на всякий случай нужно продемонстрировать эксперту, который будет проверять часть **C** вашей работы, что вы понимаете суть дела («счетчик необходимо сначала обнулить»)
4) основной цикл обработки вводимых строк можно записать на псевдокоде так:
```
for i:=1 to N do begin
 { читаем очередную строку }
 { определяем номер школы k }
 C[k] := C[k] + 1; { увеличиваем счетчик k-ой школы }
end;
```
5) поскольку данные вводятся в виде символьной строки, нужно выделить в памяти переменную **s** типа **string**
6) для чтения очередной строки будем использовать оператор **readln**
7) остается понять, как выделить из строки номер школы; по условию он закодирован в последней части строки, после второго пробела; значит, нужно найти этот второй пробел, вырезать из строки весь «хвост» после этого пробела, и преобразовать его из символьного формата в числовой
8) чтобы найти первый пробел и «отрезать» первую часть строки с этим пробелом, можно использовать команды
```
p := Pos(' ', s);
s := Copy(s, p+1, Length(s)-p);
```
первая команда определяет номер первого пробела и записывает его в целую переменную **p**, в вторая – записывает в строку **s** весь «хвост», стоящий за этим пробелом, начиная с символа с номером **p+1**; длина хвоста равна **Length(s)-p**, где **Length(s)** – длина строки;
9) поскольку нас интересует часть после **второго** пробела, эти две строчки нужно повторить два раза, в результате в переменной **s** окажется символьная запись номера школы;
10) заметим, что можно избежать дублирования двух строк, «свернув» их во внутренний цикл, но это вряд ли сильно упростит запись:
```
for k:=1 to 2 do begin
 p := Pos(' ', s);
 s := Copy(s, p+1, Length(s)-p);
end;
```
11) в пп. 8-10 описан достаточно общий метод, при котором инициалы могут быть любой длины, (но без пробела); в данном случае в условии четко сказано, что инициалы представляют собой именно 4 символа (буква, точка, буква, точка), поэтому можно найти первый пробел, а затем взять «хвост», который идет через 6 символов от него:
```
p := Pos(' ', s);
s := Copy(s,p+6,Length(s));
```
или так
```
p := Pos(' ', s);
Delete(s, 1, p+5);
```
12) для преобразования номера школы из символьного вида в числовой можно использовать функцию **Val**:

```
Val(s, k, r);
```

эта процедура (*Turbo Pascal, Borland Pascal, PascalABC,* среда *АЛГО*) преобразует символьную строку **s** в числовое значение **k**; с помощью переменной **r** обнаруживается ошибка: если раскодировать число не удалось (в строке не число), в **r** будет записан нуль (здесь мы не будем обрабатывать эту ошибку, полагая, что все данные правильные);

если вы работаете на *ПаскалеABC* (никто не может вам запретить написать, что этот так), вместо **Val** можно использовать более удобную и понятную функцию **StrToInt**:

```
k := StrToInt(s);
```

13) таким образом, основной цикл выглядит так:

```
for i:=1 to N do begin
 readln(s); { читаем очередную строку }
  { выделяем часть после второго пробела }
 p := Pos(' ', s);
 Delete(s, 1, p+5);
  { определяем номер школы k }
 Val(s, k, r);
 C[k] := C[k] + 1; { увеличиваем счетчик k-ой школы }
end;
```

14) дальше стандартным алгоритмом определяем в массиве **C** минимальный элемент **Min**, не учитывая нули (школы, из которых не было участников):

```
Min := N;
for k:=1 to 99 do
 if (C[k]<>0) and (C[k]<Min) then Min:=C[k];
```

здесь интересна первая строчка, **Min:=N**: по условию всего было **N** участников, поэтому минимальное значение не может быть больше **N**; обратите внимание, что привычный вариант (который начинается с **Min:=C[1]**) работает неверно, если из первой школы не было ни одного участника

15) и выводим на экран номера всех школ (обратите внимание – **номера!**), для которых **C[k]=Min**:

```
for k:=1 to 99 do
 if C[k]=Min then writeln(k);
```

16) остается «собрать» программу, чтобы получилось полное решение; максимальное количество школ мы задали в виде константы **LIM**:

```
const LIM = 99;
var C:array[1..LIM] of integer;
  i, p, N, k, r, Min: integer;
  s:string;
begin
readln(N);
for i:=1 to N do begin
 readln(s); { читаем очередную строку }
  { выделяем часть после второго пробела }
 p := Pos(' ', s);
 Delete(s, 1, p+5);
  { определяем номер школы k }
 Val(s, k, r);
 C[k] := C[k] + 1; { увеличиваем счетчик k-ой школы }
end;
Min := N;
for k:=1 to LIM do
```

```
      if (C[k] <> 0) and (C[k]<Min) then Min := C[k];
    for k:=1 to LIM do
     if C[k] = Min then writeln(k);
    end.
```

На что обратить внимание:

- внимательно читайте условие, убедитесь, что вы понимаете смысл каждой строчки; для каждой мелочи постарайтесь определить, зачем она добавлена в условие, что она дает для решения задачи, что ограничивает, что не разрешает делать
- определите, какая именно информация из условия нужна для решения задачи, а какая – не нужна
- определите, что именно требуется вывести на экран в результате работы программы
- начинайте составлять программу с больших блоков, записывая ее сначала на псевдокоде, а потом уточняя детали
- проверяйте «крайние» варианты (например, возможность выхода за границы массива)
- проверьте, правильно ли заданы (и заданы ли вообще) начальные значения для всех переменных
- будьте внимательны, когда в массиве есть «мертвые» элементы, которые не нужно учитывать; проверяйте, что в этом случае ваши алгоритмы (например, поиск минимального элемента) работают правильно
- проверьте, правильно ли расставлены операторные скобки **begin-end**, ограничивающие тело цикла; их обязательно нужно ставить, если в теле цикла несколько операторов
- при использовании функции **Pos** не забывайте, что первый параметр – **что** ищем (образец), а второй – **где** ищем
- чтобы эксперту было легче понять вашу программу (особенно, если она получилась «нестандартной»), пишите комментарии; объясняйте, что хранится в основных переменных
- если это возможно, желательно работать только с целыми числами; этим вы избежите проблем, связанных с округлением и неточностью хранения дробных вещественных чисел в памяти компьютера

Еще пример задания:

На вход программе подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке сообщается количество учеников N, которое не меньше 10, но не превосходит 100, каждая из следующих N строк имеет следующий формат:

<Фамилия> <Имя> <оценки>,

где <Фамилия> – строка, состоящая не более чем из 20 символов, <Имя> – строка, состоящая не более чем из 15 символов, <оценки> – через пробел три целых числа, соответствующие оценкам по пятибалльной системе. <Фамилия> и <Имя>, а также <Имя> и <оценки> разделены одним пробелом. Пример входной строки:

Иванов Петр 4 5 3

Требуется написать как можно более эффективную программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет выводить на экран фамилии и имена трех худших по среднему баллу учеников. Если среди остальных есть ученики, набравшие тот же средний балл, что и один из трех худших, то следует вывести и их фамилии и имена.

Как правильно понимать условие?

1) как и в предыдущей задаче, данные «подаются на вход программе», то есть, их можно читать с помощью операторов **read** (**readln**), предполагая, что кто-то вводит эти данные с клавиатуры вручную
2) «количество учеников не меньше 10, но не превосходит 100», здесь только вторая часть – полезная информация, она намекает на то, что придется все введенные данные одновременно держать в памяти, выделив массив (или массивы) размером 100 элементов
3) сказано, что фамилия имеет длину не более 20 символов, а имя – не более 15; здесь, по сути, важно лишь то, что фамилия и имя (вместе) занимают меньше 255 символов, то есть, «влезут» в стандартное ограничение (255 символов) для типа **string** в классических версиях Паскаля
4) после фамилии и имени записаны три оценки (а не одно число, как в прошлой задаче), причем по условию нас НЕ интересуют эти числа, а интересует только средний балл каждого ученика;
5) важно! средний балл – это вещественное число (может иметь дробную часть), тут уже стоит задуматься: все задачи обычно составляются так, чтобы они решались «хорошо», в то же время операции с дробными числами (почти) всегда выполняются с ошибками, поскольку большинство вещественных чисел нельзя точно (стандартными методами) представить в памяти реального компьютера
6) следующий шаг к правильному решению: поскольку число оценок у всех учеников одинаковое, средний балл для каждого это сумма его оценок, деленная на 3; поэтому вместо среднего балла мы можем сравнивать суммы баллов – целые числа!
7) требуется вывести фамилии и имена (баллы не нужны!) трех худших учеников, причем их может быть и больше, если несколько «худших» набрали одинаковую сумму баллов
8) если бы требовался один худший – все решается поиском по массиву; первая идея – найти самого худшего (1 проход), затем – 2-ого с конца (еще 1 проход), и, наконец, 3-его (всего три прохода по массиву)
9) это не лучший вариант (на экзамене будут сняты баллы) по двум причинам:
   - в таком методе решения три прохода по массиву, а в самом деле достаточно одного (см. далее), значит, программа неэффективна
   - непонятно, что делать в том случае, если худших – больше трех (в предельном случае – вообще все!) – за это также снимут баллы (программа работает не для всех вариантой входных данных)
10) возникает следующий вариант – отсортировать массив про возрастанию суммы (и, следовательно, среднего балла), одновременно переставляя имена и фамилии, а затем вывести самых худших, которые после сортировки окажутся в начале массива
11) этот вариант тоже плох, потому что программа неэффективна; «школьные» алгоритмы сортировки (метод «пузырька», метод выбора) имеют сложность $O(N^2)$, а надо попытаться найти метод со сложностью $O(N)$

Решение (общий подход):

1) сначала составим программу в самом общем виде на псевдокоде, чтобы определить ее основные блоки, а потом будем их постепенно «расшифровывать» через операторы языка программирования:

```
{ читаем все данные и запоминаем их }
{ находим три худших результата }
{ выводим фамилии и имена тех, чей результат меньше или
 равен «третьему худшему» }
```

2) до того, как начать писать «нормальный» код, нужно определить, как хранить данные; в данном случае нужно запомнить несколько данных по каждому ученику, их удобнее объединить в запись с двумя полями (фамилия-имя и сумма баллов); таких записей нужно выделить в памяти не менее 100 (по условию), то есть, массив из 100 элементов:

```
const LIM=100;
var Info: array[1..LIM] of record
        name: string;
        sum: integer;
      end;
```

Чтение данных:

3) после того, как мы прочитали фактическое число учеников N, в цикле считываем и расшифровываем информацию о них, сохраняя все данные в структурах

```
for i:=1 to N do begin
 { считываем строку данных }
 Info[i].name := { фамилия и имя };
 Info[i].sum := { сумма баллов };
end;
```

4) здесь, в принципе, можно использовать тот же подход, что и в первой задаче – читаем строку целиком, затем «разбираем» ее на части с помощью стандартных функций – однако, для разнообразия, мы используем другой подход – будем читать информацию **посимвольно**, то есть, считывая по одному символу в переменную **c** типа **char**;

5) сначала в поле **name** очередной структуры записываем пустую строку **''** (в которой нет ни одного символа, длина равна нулю)

```
Info[i].name := ''; { пустая строка }
```

6) затем считываем символы фамилии и сразу приписываем их в конец поля **name**:

```
repeat
 read ( c );
 Info[i].name := Info[i].name + c;
until c = ' '; { пока не прочитали пробел }
```

7) затем также читаем из входного потока имя, до пробела, и записываем его в конец того же поля **name**:

```
repeat
 read ( c );
 Info[i].name := Info[i].name + c;
until c = ' '; { пока не прочитали пробел }
```

заметьте, что эти два цикла одинаковы, поэтому ввод имени и фамилии можно записать в виде вложенного цикла так:

```
Info[i].name := ''; { пустая строка }
for k:=1 to 2 do
 repeat
  read ( c );
  Info[i].name := Info[i].name + c;
 until c = ' '; { пока не прочитали пробел }
```

8) важно! обратите внимание, что для организации внутреннего цикла используется другая переменная, **k** (а не **i**, потому что **i** – переменная главного цикла, она обозначает номер текущего ученика)

9) теперь во входном потоке остались три числа, которые мы можем последовательно считывать в целую переменную **mark**, а затем – добавлять к полю **Info[i].sum**:

```
 Info[i].sum := 0;
 for k:=1 to 3 do begin
```

```
      read(mark);
      Info[i].sum := Info[i].sum + mark;
     end;
     readln;
```

10) последняя команда **readln** пропускает все оставшиеся символы до новой строки (из этой мы прочитали все, что нужно)
11) вот полный цикл ввода данных, после его окончания все исходные данные будут записаны в первые **N** записей массива **Info**:

```
    for i:=1 to N do begin
      { ввод имени и фамилии }
     Info[i].name := '';
     for k:=1 to 2 do
      repeat
       read(c);
       Info[i].name := Info[i].name + c;
      until c = ' ';
      { ввод и суммирование оценок }
     Info[i].sum := 0;
     for k:=1 to 3 do begin
      read(mark);
      Info[i].sum := Info[i].sum + mark;
     end;
     readln;
    end;
```

Поиск трех худших данных:

12) теперь нужно придумать, как за один проход по массиву найти три худших результата;
13) как бы мы решили эту задачу, если бы нам нужно было просмотреть столбик чисел и найти три минимальных? можно сделать, например, так:
    - на бумажке вести записи в три столбика, в первом записывать минимальное число, в втором – следующее по величине, в третьем – «третье минимальное»
    - сначала пишем первое число в первый столбик, оно – минимальное, потому что других мы не еще видели; пусть это число 14:

минимум	второе	третье
14		

    - пусть следующее число – 12; оно меньше минимального, поэтому его нужно записывать в первый столбец, а «старое» минимальное число «переедет» во второй столбец

минимум	второе	третье
~~14~~		
12	14	

    - пусть дальше идет число 10 – теперь оно станет минимальным, его нужно записывать в первый столбец; при этом 12 «переедет» из первого столбца во второй, а 14 – из второго в третий

минимум	второе	третье
~~14~~		
~~12~~	~~14~~	
10	12	14

    - пусть следующее число – 11; оно больше минимального, но меньше «второго», поэтому его нужно поставить во второй столбец; число 12 из второго столбца перемещается в третий, а число 14 из третьего столбца удаляется из кандидатов в «три минимальных»

минимум	второе	третье
~~14~~		
~~12~~	~~14~~	
10	~~12~~	~~14~~
	11	12

- просмотрев таким образом весь столбик чисел, за один проход (!) можно найти три минимальных элемента
- остается только переложить этот алгоритм на язык программирования

14) выделим в памяти три целых переменных: **min1** (минимальный), **min2** («второй минимальный»), **min3** («третий минимальный»), в виде начальных значений запишем в каждую из них число, заведомо превышающее максимальную возможную сумму трех оценок, например, 20 (**>5+5+5**)
15) полный цикл поиска выглядит так:

```
min1 := 20; min2 := 20; min3 := 20;
for i:=1 to N do begin
 if Info[i].sum < min1 then begin { новый min1 }
  min3 := min2; min2 := min1;
  min1 := Info[i].sum;
 end
 else if Info[i].sum < min2 then begin { новый min2 }
   min3 := min2;
   min2 := Info[i].sum;
  end
  else if Info[i].sum < min3 then { новый min3 }
     min3 := Info[i].sum;
end;
```

16) обратим внимание на два момента: во-первых, когда переезжают два элемента, сначала нужно перемещать второй на место третьего, а потом – первый на место второго:

```
min3 := min2;
min2 := min1;
```

эти операторы нельзя менять местами, иначе «старое» значение **min2** будет потеряно;
во-вторых, если проверять условие **Info[i].sum<min2** нужно только тогда, когда очередная сумма не меньше, чем **min1**, поэтому каждый следующий условный оператор стоит в **else**-блоке предыдущего, то есть, выполняется только тогда, когда предыдущий не сработал

17) итак, мы нашли три минимальных результата, и остается вывести на экран фамилии и имена тех, у кого сумма баллов меньше или равна **min3**:

```
for i:=1 to N do
 if Info[i].sum <= min3 then
   writeln(Info[i].name);
```

18) на всякий случай приведем полную программу, она получилась довольно длинная

```
const LIM = 100;
var Info: array[1..LIM] of record
      name: string;
      sum: integer;
     end;
    i, k, N, mark, min1, min2, min3: integer;
  c: char;
begin
readln(N);
```

```
  { ввод исходных данных }
for i:=1 to N do begin
 Info[i].name := '';
 for k:=1 to 2 do
  repeat
   read(c);
   Info[i].name := Info[i].name + c;
  until c = ' ';
 Info[i].sum := 0;
 for k:=1 to 3 do begin
  read(mark);
  Info[i].sum := Info[i].sum + mark;
 end;
 readln;
end;
  { поиск трех минимальных }
min1 := 20; min2 := 20; min3 := 20;
for i:=1 to N do begin
 if Info[i].sum <min1 then begin
  min3 := min2; min2 := min1;
  min1 := Info[i].sum;
 end
 else if Info[i].sum <min2 then begin
   min3 := min2;
   min2 := Info[i].sum;
  end
  else if Info[i].sum <min3 then
     min3 := Info[i].sum;
end;
  { вывод результата }
for i:=1 to N do
 if Info[i].sum <= min3 then
   writeln(Info[i].name);
end.
```

19) эту задачу можно решить и без записей, используя два массива: массив символьных строк **name** и массив целых чисел **sum**, они объявляются так:

```
var name: array[1..MAX] of string;
  sum: array[1..MAX] of integer;
```

после этого в приведенной программе нужно заменить везде **Info[i].name** на **name** и **Info[i].sum** на **sum**.

На что обратить внимание:

- в исходных данных выделите то, что не нужно для решения задачи; при чтении эти части можно просто пропускать;
- если нам не нужны фамилия и имя отдельно, можно хранить их вместе, в виде одной строки
- если нас интересует только сумма оценок, не нужно хранить их в памяти по отдельности
- если можно при решении задачи обойтись без вещественных чисел, сделав все вычисления только с целыми числами – нужно поступить именно так (иначе снимут баллы), поскольку операции с вещественными числами во многих случаях случаев выполняются неточно

- алгоритм сложности $O(N^2)$ (например, сортировку) нужно использовать только тогда, когда нет алгоритма сложности $O(N)$; как правило, в задачах ЕГЭ такой алгоритм всегда можно (попытаться) найти; за неэффективный алгоритм при оценке решения будут сняты баллы

За что снимают баллы:

- программа работает не для всех исходных данных, не обрабатывает некоторые частные случаи
- неверно реализован алгоритм поиска минимального элемента, сортировки и т.п.
- неэффективность алгоритма:
  - используется алгоритм, имеющий сложность $O(N^2)$, когда есть алгоритм сложности $O(N)$
  - используется несколько проходов по массиву, когда достаточно одного
  - лишний расход памяти (используются дополнительные массивы или размер массива определен неверно)
  - используются операции с вещественными числами, когда можно все решить в целых числах
- переменная не описана или описана неверно
- переменным не присвоены нужные начальные значения (например, не обнуляются счетчики) или присвоены неверные значения
- нет вывода результата в конце программы
- перепутаны знаки < и >, логические операции **`or`** и **`and`**
- применяется недопустимая операция, например, **`div`** или **`mod`** для вещественных чисел
- неверно расставлены операторные скобки **`begin-end`**
- в цикле **`for`** используется вещественная переменная (Паскаль)
- в цикле **`while`** или **`repeat`** не изменяется переменная цикла, из-за чего происходит зацикливание
- синтаксические ошибки (знаки пунктуации – запятые, точки, точки с запятой; неверное написание ключевых слов); чтобы получить 4 балла, при абсолютно верном решении нужно сделать не более одной синтаксической ошибки; на 3 балла – до трех ошибок, на 2 балла – до пяти и на 1 балл – до семи ошибок

Задачи для тренировки[2]:

1) На вход программы подается 366 строк, которые содержат информацию о среднесуточной температуре всех дней 2008 года. Формат каждой из строк следующий: сначала записана дата в виде dd.mm (на запись номера дня и номера месяца в числовом формате отводится строго два символа, день от месяца отделен точкой), затем через пробел записано значение температуры — число со знаком плюс или минус, с точностью до 1 цифры после десятичной точки. Данная информация отсортирована по значению температуры, то есть хронологический порядок нарушен. Требуется написать программу на языке Паскаль или Бейсик, которая будет выводить на экран информацию о месяце (месяцах), среднемесячная температура у которого (которых) наименее отклоняется от среднегодовой. В первой строке вывести среднегодовую температуру. Найденные значения для каждого из месяцев следует выводить в отдельной строке в виде: номер месяца, значение среднемесячной температуры, отклонение от среднегодовой температуры.

2) На вход программы подается содержится текст на английском языке, заканчивающийся точкой (другие символы "." в тексте отсутствуют). Требуется написать программу, которая будет определять и выводить на экран английскую букву, встречающуюся в этом тексте чаще всего, и количество там таких букв. Строчные и прописные буквы при этом считаются не различимыми. Если искомых букв несколько, то программа должна выводить на экран первую из них по алфавиту. Например, пусть файл содержит следующую запись:

   `It is not a simple task. Yes!`

   Чаще всего здесь встречаются буквы **`I`**, **`S`** и **`T`** (слово **`Yes`** в подсчете не учитывается, так как расположено после точки). Следовательно, в данном случае программа должна вывести два символа, разделенных пробелом: **`I 3`**

3) На вход программы подаются произвольные алфавитно-цифровые символы. Ввод этих символов заканчивается точкой. Требуется написать программу, которая будет печатать последовательность строчных английских букв ('a' 'b'... 'z') из входной последовательности и частот их повторения. Печать должна происходить в алфавитном порядке. Например, пусть на вход подаются следующие символы:

   `fhb5kbfыshfm.`

   В этом случае программа должна вывести

   `b2`
   `f3`
   `h2`
   `kl`
   `ml`
   `sl`

[2] Источники заданий:

1. Демонстрационные варианты ЕГЭ разных лет.
2. Гусева И.Ю. ЕГЭ. Информатика: раздаточный материал тренировочных тестов. — СПб: Тригон, 2009.
3. Самылкина Н.Н., Островская Е.М. Информатика: тренировочные задания. – М.: Эксмо, 2009.
4. Зорина Е.М., Зорин М.В. ЕГЭ-2010: Информатика: Сборник заданий. – М.: Эксмо, 2009.
5. Якушкин П.А., Крылов С.С. ЕГЭ-2010. Информатика: сборник экзаменационных заданий. – М.: Эксмо, 2009.
6. Якушкин П.А., Ушаков Д.М. Самое полное издание типовых вариантов реальных заданий ЕГЭ 2010. Информатика. — М.: Астрель, 2009.
7. Тренировочные и диагностические работы МИОО.

4) На вход программы подаются фамилии и имена учеников. Известно, что общее количество учеников не превосходит 100. В первой строке вводится количество учеников, принимавших участие в соревнованиях, N. Далее следуют N строк, имеющих следующий формат:

<Фамилия> <Имя>

Здесь *<Фамилия>* – строка, состоящая не более чем из 20 символов; *<Имя>* – строка, состоящая не более чем из 15 символов. При этом *<Фамилия>* и *<Имя>* разделены одним пробелом. Примеры входных строк:

Иванова Мария
Петров Сергей

Требуется написать программу, которая формирует и печатает уникальный логин для каждого ученика по следующему правилу: если фамилия встречается первый раз, то логин – это данная фамилия, если фамилия встречается второй раз, то логин – это фамилия, в конец которой приписывается число 2 и т.д. Например, для входной последовательности

Иванова Мария
Петров Сергей
Бойцова Екатерина
Петров Иван
Иванова Наташа

будут сформированы следующие логины:

Иванова
Петров
Бойцова
Петров2
Иванова2

5) На городской олимпиаде по информатике участникам было предложено выполнить 3 задания, каждое из которых оценивалось по 25-балльной шкале. Известно, что общее количество участников первого тура олимпиады не превосходит 250 человек. На вход программы подаются сведения о результатах олимпиады. В первой строке вводится количество участников N. Далее следуют N строк, имеющих следующий формат:

<Фамилия> <Имя> <Баллы>

Здесь *<Фамилия>* – строка, состоящая не более чем из 20 символов; *<Имя>* – строка, состоящая не более чем из 15 символов; *<Баллы>* – строка, содержащая три целых числа, разделенных пробелом, соответствующих баллам, полученным участником за каждое задание первого тура. При этом *<Фамилия>* и *<Имя>*, *<Имя>* и *<Баллы>* разделены одним пробелом. Примеры входных строк:

Петрова Ольга 25 18 16
Калиниченко Иван 14 19 15

Напишите программу, которая будет выводить на экран фамилию и имя участника, набравшего максимальное количество баллов. Если среди остальных участников есть ученики, набравшие такое же количество баллов, то их фамилии и имена также следует вывести. При этом имена и фамилии можно выводить в произвольном порядке.

6) На вход программы подаются сведения о результатах соревнований по школьному многоборью. Многоборье состоит из соревнований по четырем видам спорта, участие в каждом из которых оценивается баллами от 0 до 10 (0 баллов получает ученик, не принимавший участия в соревнованиях по данному виду спорта). Победители определяются по наибольшей сумме набранных баллов. Известно, что общее количество участников соревнований не превосходит 100.

В первой строке вводится количество учеников, принимавших участие в соревнованиях, N. Далее следуют N строк, имеющих следующий формат:

<Фамилия> <Имя> <Баллы>

Здесь *<Фамилия>* – строка, состоящая не более чем из 20 символов; *<Имя>* – строка, состоящая не более чем из 15 символов; *<Баллы>* - строка, содержащая четыре целых числа, разделенных пробелом, соответствующих баллам, полученным на соревнованиях по каждому из четырех видов спорта. При этом *<Фамилия>* и *<Имя>*, *<Имя>* и *<Баллы>* разделены одним пробелом. Примеры входных строк:

Иванова Мария 5 8 6 3
Петров Сергей 9 9 5 7

Напишите программу, которая будет выводить на экран фамилии и имена трех лучших участников многоборья. Если среди остальных участников есть ученики, набравшие то же количество баллов, что и один из трех лучших, то их фамилии и имена также следует вывести. При этом имена и фамилии можно выводить в произвольном порядке.

7) В некотором вузе абитуриенты проходят предварительное тестирование, по результатам которого могут быть допущены к сдаче вступительных экзаменов в первом потоке. Тестирование проводится по двум предметам, по каждому предмету абитуриент может набрать от 0 до 100 баллов. При этом к сдаче экзаменов в первом потоке допускаются абитуриенты, набравшие по результатам тестирования не менее 30 баллов по каждому из двух предметов. На вход программы подаются сведения о результатах предварительного тестирования. Известно, что общее количество участников тестирования не превосходит 500.
В первой строке вводится количество абитуриентов, принимавших участие в тестировании, N. Далее следуют N строк, имеющих следующий формат:

<Фамилия> <Имя> <Баллы>

Здесь *<Фамилия>* – строка, состоящая не более чем из 20 символов; *<Имя>* – строка, состоящая не более чем из 15 символов; *<Баллы>* – строка, содержащая два целых числа, разделенных пробелом, соответствующих баллам, полученным на тестировании по каждому из двух предметов. При этом *<Фамилия>* и *<Имя>*, *<Имя>* и *<Баллы>* разделены одним пробелом. Примеры входных строк:

Ветров Роман 68 59
Анисимова Екатерина 64 88

Напишите программу, которая будет выводить на экран фамилии и имена абитуриентов, потерпевших неудачу, то есть не допущенных к сдаче экзаменов в первом потоке. При этом фамилии должны выводиться в алфавитном порядке.

8) На вход программе подаются сведения о телефонах всех сотрудников некоторого учреждения. В первой строке сообщается количество сотрудников N, каждая из следующих N строк имеет следующий формат:

<Фамилия> <Инициалы> <телефон>

где *<Фамилия>* – строка, состоящая не более чем из 20 символов, *<Инициалы>* - строка, состоящая не более чем из 4-х символов (буква, точка, буква, точка), *<телефон>* – семизначный номер, 3-я и 4, я, а также 5-я и 6-я цифры которого разделены символом «–». *<Фамилия>* и *<Инициалы>*, а также *<Инициалы>* и *<телефон>* разделены одним пробелом. Пример входной строки:

Иванов П.С. 555-66-77

Сотрудники одного подразделения имеют один и тот же номер телефона. Номера телефонов в учреждении отличаются только двумя последними цифрами. Требуется написать как можно более эффективную программу, которая будет выводить на экран информацию, сколько в среднем сотрудников работает в одном подразделении данного учреждения.

9) На вход программе сначала подается число участников олимпиады **N**. В каждой из следующих **N** строк находится результат одного из участников олимпиады в следующем формате:

<Фамилия> <Имя> <класс> <баллы>

где *<Фамилия>* – символьная строка (не более 20 символов), *<Имя>* – символьная строка (не более 15 символов), *<класс>* – число от 7 до 11, *<баллы>* – целое число набранных участником баллов. *<Фамилия>* и *<Имя>*, *<Имя>* и *<класс>*, а также *<класс>* и *<баллы>* разделены одним пробелом. Пример входной строки:

Семенов Егор 11 225

Победителем олимпиады становится участник, набравший наибольшее количество баллов, при условии, что он набрал более 200 баллов. Если такое количество баллов набрали несколько участников, то все они признаются победителями при выполнении условия, что их доля не превышает 20% от общего числа участников.

Победителем олимпиады не признается никто, если нет участников, набравших больше 200 баллов, или больше 20% от общего числа участников набрали одинаковый наибольший балл.

Напишите эффективную по времени работы и по используемой памяти программу, которая будет определять фамилию и имя лучшего участника, не ставшего победителем олимпиады. Если таких участников несколько, т.е. если следующий за баллом победителей один и тот же балл набрали несколько человек, или, если победителей нет, а лучших участников несколько (в этом случае именно они являются искомыми), то выдается только количество искомых участников. Гарантируется, что искомые участники (участник) имеются.

Программа должна выводить через пробел фамилию и имя искомого участника или их количество. Пример выходных данных (один искомый участник):

Семенов Егор

Второй вариант выходных данных (несколько искомых участников):

12

10) В молочных магазинах города X продается сметана с жирностью 15, 20 и 25 процентов. В городе X был проведен мониторинг цен на сметану. Напишите эффективную по времени работы и по используемой памяти программу, которая будет определять для каждого вида сметаны, сколько магазинов продают ее дешевле всего. На вход программе сначала подается число магазинов N. В каждой из следующих N строк находится информация в следующем формате:

<Фирма> <Улица> <Жирность> <Цена>

где *<Фирма>* – строка, состоящая не более, чем из 20 символов без пробелов, *<Улица>* – строка, состоящая не более, чем из 20 символов без пробелов, <Жирность> – одно из чисел – 15, 20 или 25, *<Цена>* – целое число в диапазоне от 2000 до 5000, обозначающее стоимость одного литра сметаны в копейках. *<Фирма>* и *<Улица>, <Улица>* и *<Жирность>*, а также *<Жирность>* и *<Цена>* разделены ровно одним пробелом. Пример входной строки:

Перекресток Короленко 25 3200

Программа должна выводить через пробел 3 числа – количество магазинов, продающих дешевле всего сметану с жирностью 15, 20 и 25 процентов. Если какой-то вид сметаны нигде не продавался, то следует вывести 0.

Пример выходных данных:

12 10 0

11) Школьная олимпиада по информатике проводилась для учеников 7-11-х классов, участвующих в общем конкурсе. Каждый участник олимпиады мог набрать от 0 до 70 баллов. Для определения призеров олимпиады сначала отбираются 25% участников, показавших лучшие результаты. Если у последнего участника, входящего в 25%, оказывается такое же количество баллов, как и у следующих за ним в итоговой таблице, все они считаются призерами только тогда, когда

набранные ими баллы больше половины максимально возможных; иначе все они не считаются призерами.

Напишите эффективную по времени работы и по используемой памяти программу, которая по результатам олимпиады будет определять минимальный балл призера олимпиады, и количество призеров было в каждой параллели (среди 7-х, 8-х, 9-х, 10-х и 11-х классов отдельно). Гарантируется, что хотя бы одного призера по указанным правилам определить можно.

На вход программе сначала подается число участников олимпиады N. В каждой из следующих N строк находится результат одного из участников олимпиады в следующем формате:

<Фамилия> <Имя> <класс> <баллы>

где *<Фамилия>* – строка, состоящая не более, чем из 30 символов, *<Имя>* – строка, состоящая не более, чем из 15 символов, *<класс>* – число от 7 до 11, *<баллы>* – целое число от 0 до 70 набранных участником баллов. *<Фамилия>* и *<Имя>*, *<Имя>* и *<класс>*, а также *<класс>* и *<баллы>* разделены одним пробелом. Пример входной строки:

Семенов Сидор 11 66

Программа должна выводить в первой строке минимальный балл призера, а в следующей – число призеров по всем параллелям отдельно.

Пример выходных данных:

63
1 5 8 12 22

12) В некотором вузе абитуриенты проходили предварительное тестирование, по результатам которого они могут быть допущены к сдаче вступительных экзаменов в первом потоке. Тестирование проводится по двум предметам, по каждому предмету абитуриент может набрать от 0 100 баллов. При этом к сдаче экзаменов в первом потоке допускаются абитуриенты, набравшие по результатам тестирования не менее 30 баллов по каждому из двух предмета. На вход программы подаются сведения о результатах предварительного тестирования. Известно, что общее количество участников тестирования не превосходит 500.

В первой строке вводится количество абитуриентов, принимавших участие в тестировании, **N**. Далее следуют **N** строк, имеющих следующий формат:

<Фамилия> <Имя> <Баллы>

Здесь *<Фамилия>* – строка, состоящая не более чем из 20 символов; *<Имя>* – строка, состоящая не более чем из 15 символов, *<Баллы>* – строка, содержащая два целых числа, разделенных пробелом – баллы, полученные на тестировании по каждому из двух предметов. При этом *<Фамилия>* и *<Имя>*, *<Имя>* и *<Баллы>* разделены одним пробелом. Пример входной строки:

Романов Вельямин 48 39

Напишите программу, которая будет выводить на экран фамилии и имена абитуриентов, потерпевших неудачу, то есть не допущенных к сдаче экзаменов в первом потоке. При этой фамилии должны выводиться в алфавитном порядке.

13) На автозаправочных станциях (АЗС) продается бензин с маркировкой 92, 95 и 98. В городе **N** был проведен мониторинг цены бензина на различных АЗС. Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет определять для каждого вида бензина, сколько АЗС продают его дешевле всего.

На вход программе в первой строке подается число данных **N** о стоимости бензина. В каждой из последующих **N** строк находится информация в следующем формате:

<Компания> <Улица> <Марка> <Цена>

где *<Компания>* – строка, состоящая не более, чем из 20 символов без пробелов, *<Улица>* – строка, состоящая не более, чем из 20 символов без пробелов, *<Марка>* – одно из чисел – 92, 95

или 98, <*Цена*> – целое число в диапазоне от 1000 до 3000, обозначающее стоимость одного литра бензина в копейках.

<*Компания*> и <*Улица*>, <*Улица*> и <*Марка*>, а также <*Марка*> и <*Цена*> разделены ровно одним пробелом. Пример входной строки:

Синойл Цветочная 95 2250

Программа должна выводить через пробел 3 числа – количество АЗС, продающих дешевле всего 92-й, 95-й и 98-й бензин соответственно. Если бензин какой-то марки нигде не продавался, то следует вывести 0.

Пример выходных данных:

12 1 0

14) На вход программы подаются прописные латинские буквы, ввод этих символов заканчивается точкой. Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет определять, можно ли переставить эти буквы так, чтобы получился палиндром (палиндром читается одинаково слева направо и справа налево). Программа должна вывести ответ «Да» или «Нет», а в случае ответа «Да» – еще и сам полученный палиндром (первый в алфавитном порядке).

Пример входной строки:

GAANN

Пример выходных данных:

Да
ANGNA

15) В соревнованиях по многоборью (из M видов спорта) участвуют N спортсменов (N < 1000) . На вход программе в первой строке подается число спортсменов N, во второй – число видов спорта M. В каждой из последующих N строк находится информация в следующем формате:

<Фамилия> <Имя> <Баллы>

где <*Фамилия*> – строка, состоящая не более, чем из 20 символов без пробелов, <*Имя*> – строка, состоящая не более, чем из 12 символов без пробелов, <*Баллы*> – M целых чисел, обозначающие количество баллов, набранных спортсменом в каждом из видов многоборья.

<*Фамилия*> и <*Имя*>, <Имя> и <Баллы>, а также отдельные числа в поле <*Баллы*> разделены ровно одним пробелом. Пример входных строк:

3
4
Иванов Сергей 100 30 78 13
Петров Антон 90 16 98 14
Сидоров Юрий 100 70 30 21

Программа должна выводить результирующую таблицу, содержащую список спортсменов, отсортированный по убыванию суммы баллов, набранные суммы и занятые места.

В данном случае программа должна вывести

Иванов Сергей 221 1
Сидоров Юрий 221 1
Петров Антон 218 2

16) На вход программе подаются сведения о пассажирах, сдавших свой багаж в камеру хранения. В первой строке задано текущее время: через двоеточие два целых числа, соответствующие часам (от 00 до 21, ровно 2 символа) и минутам (от 00 до 59, ровно 2 символа). Во второй строке задается количество пассажиров N, которое не меньше 10, но не превосходит 1000. В каждой из последующих N строк находится информация о пассажирах в следующем формате:

<Фамилия> <Время освобождения ячейки>

где <*Фамилия*> – строка, состоящая не более, чем из 20 символов без пробелов, <*Время освобождения ячейки*> – через двоеточие два целых числа, соответствующие часам (от 00 до 21, ровно 2 символа) и минутам (от 00 до 59, ровно 2 символа). <*Фамилия*> и <*Время освобождения ячейки*> разделены ровно одним пробелом. Пример входных строк:

```
10:00
3
Иванов 12:00
Петров 10:12
Сидоров 12:12
```

Программа должна выводить список пассажиров, которые в ближайшие 2 часа должны освободить ячейки. Список должен быть отсортирован в хронологическом порядке освобождения ячеек. В данном случае программа должна вывести

```
Петров
Иванов
```

17) На вход программе подается текст заклинания, состоящего не более, чем из 200 символов, заканчивающийся точкой (другие точки во входных данных отсутствуют). Гарри Поттеру нужно зашифровать его следующим образом. Сначала Гарри определяет количество букв в самом коротком слове, обозначив полученное число через K (словом называется непрерывная последовательность английских букв, слова друга от друга отделяются любыми другими символами, длина слова не превышает 20 символов). Затем он заменяет каждую английскую букву в заклинании на букву, стоящую в английском алфавите на K букв ранее (алфавит считается циклическим, то есть, перед буквой A стоит буква Z), оставив другие символы неизменными. Строчные буквы при этом остаются строчными, а прописные – прописными.

Требуется написать программу для Гарри Поттера, которая будет выводить на экран текст зашифрованного заклинания. Например, если исходный текст был таким:

```
Zb Ra Ca Dab Ra.
```

то результат шифровки должен быть следующий:

```
Xz Py Ay Byz Py.
```

18) Имеется список результатов голосования избирателей за несколько партий, в виде списка названий данных партий. На вход программе в первой строке подается количество избирателей в списке N. В каждой из последующих N строк записано название партии, за которую проголосовал данный избиратель, в виде текстовой строки. Длина строки не превосходит 50 символов, название может содержать буквы, цифры, пробелы и прочие символы.

Пример входных данных:

```
6
Party one
Party two
Party three
Party three
Party two
Party three
```

Программа должна вывести список всех партий, встречающихся в исходном списке, в порядке убывания количества голосов, отданных за эту партию. При этом название каждой партии должно быть выведено ровно один раз, вне зависимости от того, сколько голосов было отдано за данную партию. Пример выходных данных для приведенного выше примера входных данных:

```
Party three
Party two
Party one
```

При этом следует учитывать, что количество голосов избирателей в исходном списке может быть велико (свыше 1000), а количество различных партий в этом списке не превосходит 10.

19) Имеется список людей с указанием их фамилии, имени и даты рождения. Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет определять самого старшего человека из этого списка и выводить его фамилию, имя и дату рождения, а если имеется несколько самых старших людей с одинаковой датой рождения, то определять их количество.

На вход программе в первой строке подается количество людей в списке N. В каждой из последующих N строк находится информация в следующем формате:

<Фамилия> <Имя> <Дата рождения>

где *<Фамилия>* – строка, состоящая не более, чем из 20 символов без пробелов, *<Имя>* – строка, состоящая не более, чем из 20 символов без пробелов, *<Дата рождения>* – строка, имеющая вид ДД.ММ.ГГГГ, где ДД – двузначное число от 01 до 31, ММ – двузначное число от 01 до 12, ГГГГ – четырехзначное число от 1800 до 2100.

Пример входной строки:

Иванов Сергей 27.03.1993

Программа должна вывести фамилию и имя самого старшего человека в списке.

Пример выходных данных:

Иванов Сергей

Если таких людей, несколько, то программа должна вывести их количество. Пример вывода в этом случае:

3

20) Имеется список учеников разных школ, сдававших экзамен по информатике, с указанием их фамилии, имени, школы и набранного балла. Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет определять двух учеников школы № 50, которые лучше всех сдали информатику, и выводить на экран их фамилии и имена.

Если наибольший балл набрали более двух человек, нужно вывести только их количество. Если наибольший балл набрал один человек, а следующий балл набрало несколько человек, нужно вывести только фамилию и имя лучшего. Известно, что информатику сдавали не менее 5 учеников школы № 50.

На вход программе в первой строке подается количество учеников списке N. В каждой из последующих N строк находится информация в следующем формате:

<Фамилия> <Имя> <Школа> <Балл>

где *<Фамилия>* – строка, состоящая не более, чем из 20 символов без пробелов, *<Имя>* – строка, состоящая не более, чем из 20 символов без пробелов, *<Школа>* – целое число от 1 до 99, *<Балл>* – целое число от 1 до 100.

Пример входной строки:

Иванов Сергей 50 87

Пример выходных данных, когда найдено два лучших:

Иванов Сергей
Сергеев Иван

Если больше двух учеников набрали высший балл, то программа должна вывести их количество. Пример вывода в этом случае:

8

Если высший балл набрал один человек, а следующий балл набрало несколько человек, то программа должна вывести только фамилию и имя лучшего. Пример вывода в этом случае:

Иванов Сергей

21) Имеется список учеников разных школ, сдававших экзамен по информатике, с указанием их фамилии, имени, школы и набранного балла. Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет определять номера школ, в которых средний балл выше, чем средний по району. Если такая школа одна, нужно вывести и средний балл (в следующей строчке). Известно, что информатику сдавали не менее 5 учеников. Кроме того, школ с некоторыми номерами не существует.

На вход программе в первой строке подается количество учеников списке N. В каждой из последующих N строк находится информация в следующем формате:

<Фамилия> <Имя> <Школа> <Балл>

где *<Фамилия>* – строка, состоящая не более, чем из 20 символов без пробелов, *<Имя>* – строка, состоящая не более, чем из 20 символов без пробелов, *<Школа>* – целое число от 1 до 99, *<Балл>* – целое число от 1 до 100.

Пример входной строки:

Иванов Сергей 50 87

Пример выходных данных, когда найдено три школы:

50 87 23

Пример вывода в том случае, когда найдена одна школа:

18
Средний балл = 85

22) На вход программе подается строка (длиной не более 200 символов), в которой нужно зашифровать все английские слова (словом называется непрерывная последовательность английских букв, слова друга от друга отделяются любыми другими символами, длина слова не превышает 20 символов). Строка заканчивается символом #, других символов # в строке нет. Каждое слово зашифровано с помощью циклического сдвига на длину этого слова. Например, если длина слова равна K, каждая буква в слове заменяется на букву, стоящую в английском алфавите на K букв дальше (алфавит считается циклическим, то есть, за буквой Z стоит буква A). Строчные буквы при этом остаются строчными, а прописные – прописными. Символы, не являющиеся английскими буквами, не изменяются.

Требуется написать программу, которая будет выводить на экран текст зашифрованного сообщения. Например, если исходный текст был таким:

Day, mice. "Year" is a mistake#

то результат шифровки должен быть следующий:

Gdb, qmgi. "Ciev" ku b tpzahrl#

23) После единых выпускных экзаменов по информатике в район пришла информация о том, какой ученик какой школы сколько баллов набрал. По положению об экзамене каждый район сам определяет, за какой балл нужно поставить какую оценку.

Районный методист решила, что оценку «отлично» должны получить 20% участников (целое число, с отбрасыванием дробной части). Для этого она должна определить, какой балл должен был набрать ученик, чтобы получить «отлично». Если невозможно определить такой балл, чтобы «отлично» получили ровно 20% участников, «отлично» должно получить меньше участников, чем 20%. Если таких участников не окажется (наибольший балл набрали больше 20% участников) — эти и только эти ученики должны получить «отлично».

Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например Borland Pascal 7.0), которая должна вывести на экран наименьший балл, который набрали участники, получившие «отлично». Известно, что информатику сдавало больше 5-ти учеников. Также известно, что есть такое количество баллов, которое не получил ни один участник.

На вход программе сначала подаётся число учеников, сдававших экзамен. В каждой из следующих N строк находится информация об учениках в формате:

<Фамилия> <Имя> <Номер школы> <Количество баллов>

где <*Фамилия*> — строка, состоящая не более чем из 30 символов без пробелов, <*Имя*> — строка, состоящая не более, чем из 20 символов без пробелов, <*Номер школы*> — целое число в диапазоне от 1 до 99, <*Количество баллов*> — целое число в диапазоне от 1 до 100. Эти данные записаны через пробел, причём ровно один между каждой парой (то есть, всего по три пробела в каждой строке).

Пример входной строки:

Иванов Иван 50 87

Пример выходных данных:

78

24) На вход программе подается последовательность символов, заканчивающаяся точкой. Требуется написать программу, которая определяет, есть ли в этой последовательности десятичные цифры, и выводит наибольшее число, которое можно составить из этих цифр. Ведущих нулей в числе быть не должно (за исключением числа 0, запись которого содержит ровно одну цифру). Если цифр нет, программа должна вывести на экран слово «Нет», а если есть – слово «Да» и в следующей строчке искомое число. Например, если исходная последовательность была такая:

Day 10, mice 8: "Year" 7 is a mistake 91.

то результат должен быть следующий:

Да
987110

25) На вход программе подается последовательность цифр, заканчивающаяся точкой (другие символы, кроме цифр и точки, отсутствуют). Требуется написать программу, которая выводит цифры, встречающиеся во входной последовательности, в порядке увеличения частоты их встречаемости. Если какие-то цифры встречаются одинаковое число раз, они выводятся в порядке возрастания. Например, если исходная последовательность была такая:

123124456.

то результат должен быть следующий:

356124

26) На вход программе подается последовательность символов, заканчивающаяся нулем. Ноль в этой последовательности единственный, среди символов обязательно есть другие десятичные цифры. Требуется написать программу, которая составляет из этих цифр число-палиндром максимальной длины (которое читается одинаково слева направо и справа налево). Если таких чисел несколько, нужно вывести минимальное из них. Нулей в числе быть не должно (ноль – признак окончания ввода). Все имеющиеся цифры использовать не обязательно, но количество цифр в ответе должно быть максимально возможным. Например, если исходная последовательность была такая:

for i:=99921 downto 20

то результат должен быть следующий:

29192

27) На вход программе подается предложение на английском языке, заканчивающееся точкой. Требуется написать программу, которая определяет, можно ли переставить английские буквы этого предложения (не учитывая остальные символы) так, чтобы получить палиндром – слово, которое читается одинаково слева направо и справа налево. Строчные и прописные буквы не различаются. Если это сделать нельзя, программа должна вывести на экран слово «Нет», а если можно – слово «Да» и в следующей строчке искомое слово прописными буквами. Если таких слов несколько, нужно вывести *последнее* в алфавитном порядке слово. Например, если исходная последовательность была такая:

Deed daad N.

то результат должен быть следующий:

```
Да
EDDANADDE
```

28) На вход программе подается набор символов, заканчивающийся точкой. Напишите эффективную, в том числе и по используемой памяти, программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая сначала будет определять, есть ли в этом наборе символы, соответствующие десятичным цифрам. Если такие символы есть, то можно ли переставить их так, чтобы полученное число было симметричным (читалось одинаково как слева направо, так и справа налево). Ведущих нулей в числе быть не должно, исключение – число 0, запись которого содержит ровно один ноль. Если требуемое число составить невозможно, то программа должна вывести на экран слово "NO". А если возможно, то в первой строке следует вывести слово "YES", а во второй – искомое симметричное число. Если таких чисел несколько, то программа должна выводить максимальное из них. Например, пусть на вход подаются следующие символы:

```
Do not 911 to 09 do.
```

В данном случае программа должна вывести

```
YES
91019
```

29) На вход программе подается последовательность символов, среди которых могут быть и цифры, заканчивающаяся точкой. Требуется написать программу, которая составляет и выводит минимальное число из тех цифр, которые не встречаются во входных данных. Ноль не используется. Если во входных данных встречаются все цифры от 1 до 9, то следует вывести «0». Например, если исходная последовательность была такая:

```
1A734B39.
```

то результат должен быть следующий:

```
2568
```

30) На вход программе подаются сведения о ячейках камеры хранения багажа. В первой строке – текущая дата – день (ровно две цифры, от 01 до 31), затем через точку – месяц (ровно две цифры, от 01 до 12). Во второй строке сообщается количество занятых ячеек N (не меньше 3, но не больше 1000). Каждая из следующих строк имеет формат:

```
<номер ячейки> <дата сдачи багажа>
```

Номер ячейки – это целое число, дата – 5 символов: день (ровно две цифры, от 01 до 31), затем через точку – месяц (ровно две цифры, от 01 до 12). Сведения отсортированы по номерам ячеек. Все даты относятся к одному календарному году. Считать, что в феврале 28 дней.

Нужно вывести номера тех ячеек, в которых багаж хранится более 3 дней в хронологическом порядке сдачи багажа. Например, если исходные данные были такие:

```
04.06
3
1000 01.06
1001 31.05
2007 21.05
```

то результат должен быть следующий:

```
2007
1001
```

31) На вход программе подаются сведения о студентах некоторого вуза. В первой строке сообщается количество студентов N (не более 100). Каждая из следующих строк имеет формат:

```
<фамилия> <имя> <курс> <стипендия>
```

Все данные в строке разделяются одним пробелом. Фамилия состоит не более, чем из 20 символов, имя – не более, чем из 15. Курс – целое число от 1 до 5, стипендия – целое число.

Требуется написать программу, которая будет выводить фамилии и имена всех студентов, имеющих максимальные стипендии на каждом курсе.

Пример входных строк:

```
25
Федорова Ирина 5 4500
Семенов Илья 3 2800
```

Пример выходных строк:

```
Курс 1
Петров Иван
Иванов Сидор
Курс 3
Смирнов Максим
```

32) Некоторый поезд в пути следования останавливается на N станциях (станция номер 1 — начальная, а станция номер N — конечная). Дан список пассажиров поезда, для каждого из которых известно, на какой станции он садится, а на какой — выходит. Напишите эффективную по времени работы и используемой памяти программу, которая по этим данным определяет, на каких перегонах (то есть между какими соседними станциями) в поезде было наименьшее число пассажиров. На вход программе в первой сроке подается количество станций N и количество пассажиров P. В каждой из последующих P строк находится информация о пассажирах в следующем формате:

<Фамилия> <Имя> <станция посадки> <станция выхода>

где <Фамилия> – строка, состоящая не более, чем из 20 символов без пробелов, <Имя> – строка, состоящая не более, чем из 20 символов без пробелов, <станция посадки> и <станция выхода> — числа от 1 до N, при этом номер станции посадки меньше номера станции выхода.

Пример входных данных:

```
6 3
Иванов Сергей 2 4
Сергеев Петр 1 3
Петров Кирилл 3 6
```

Программа должна вывести список перегонов, на которых в поезде было наименьшее число пассажиров. Каждый перегон выводится в виде двух последовательных номеров станций, разделенных знаком “-“. Для примера выше результат работы программы должен быть таким (на данных перегонах в поезде находилось наименьшее число пассажиров):

```
1-2
4-5
5-6
```

При выполнении задания следует учитывать, что значение N не превосходит 10, а значение P может быть большим (до 1000).

33) Дан список результатов сдачи экзамена учащимися школ некоторого района, с указанием фамилии и имени учащегося, номера школы и итогового балла. Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая определяет номера школ, в которых больше всего учащихся получило за экзамен максимальный балл среди всех учащихся района. На вход программе в первой строке подается количество учащихся во всех школах района N. В каждой из последующих N строк находится информация в следующем формате:

<Фамилия> <Имя> <Номер школы> <Балл>

где < Фамилия > - строка, состоящая не более, чем из 20 символов без пробелов, <Имя> - строка, состоящая не более, чем из 20 символов без пробелов, <Номер школы> - число от 1 до 99, <Балл> – число от 0 до 100. Порядок следования строк - произвольный.

Пример входных данных:

```
6
Иванов Сергей 7 74
Сергеев Петр 3 82
Петров Кирилл 7 85
Кириллов Егор 3 82
Егоров Николай 7 85
Николаев Иван 19 85
```

Программа должна вывести номера школ, из которых наибольшее количество учащихся получило на экзамене максимальный балл среди всех учащихся района. Пример вывода для приведенного выше примера ввода:

```
7
```

Примечание. В данном примере максимальный балл по району равен 85, его набрало 2 учащихся из школы 7 и 1 учащийся из школы 19, поэтому выводится только номер школы 7.

При выполнении задания следует учитывать, что значение N может быть велико (до 10.000).

34) На вход программе подается текстовый файл, в первой строке которого записано квадратное уравнение, причем используются обозначения:

x^2 обозначается как «a»

x обозначается как «b»

Например, уравнение $2x^2 - 4x - 6 = 0$ запишется в виде строки

```
2a-4b-6
```

Гарантируется, что уравнение имеет «хороший» вид, все его коэффициенты определены и корни вещественные. Напишите эффективную по времени работы и по используемой памяти программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая дописывает в конец файла корни уравнения. Для приведенного входного файла программа должна дописать в конец файла

```
-1
3
```

35) В командных олимпиадах по программированию для решения предлагается не более 12 задач. Команда может решать предложенные задачи в любом порядке. Подготовленные решения команда посылает в единую проверяющую систем соревнований. Вам предлагается написать эффективную, в том числе и по используемой памяти, программу, которая будет статистически обрабатывать пришедшие запросы на проверку, чтобы определить популярность той или иной задачи. Следует учитывать, что количество запросов в списке может быть очень велико, например, когда олимпиада проводится через Интернет. перед текстом программы кратко опишите используемый вами алгоритм решения задачи. На вход программе в первой строчке подается количество пришедших запросов N. В каждой из последующих N строк записан номер задачи от 1 до 12. Пример входных данных:

```
6
1
2
1
1
5
2
```

Программа должна вывести список всех задач, встречающихся в запросах, в порядке возрастания (неубывания) количества запросов по ней с указанием этого количества запросов. Каждая задача должна быть выведена только один раз. Пример выходных данных для приведенных входных данных:

```
5 1
2 2
1 3
```

36) Популярная газета объявила конкурс на выбор лучшего фильма, для которого стоит снять продолжение. На выбор читателей было предложено 10 фильмов. Вам предлагается написать эффективную, в том числе и по используемой памяти, программу, которая будет статистически обрабатывать результаты sms-голосования по этому вопросу, чтобы определить популярность того или иного фильма. Следует учитывать, что количество голосов в списке может быть очень велико. На вход программе в первой строчке подается количество пришедших sms-сообщений N. В каждой из последующих N строк записано название фильма. Пример входных данных:

```
6
Белое солнце пустыни
Бриллиантовая рука
Белое солнце пустыни
Белое солнце пустыни
Гараж
Бриллиантовая рука
```

Программа должна вывести список всех фильмов, встречающихся в списке, в порядке убывания (невозрастания) количества отданных за них голосов с указанием этого количества голосов. Название каждого фильма должно быть выведено только один раз. Пример выходных данных для приведенных входных данных:

```
Белое солнце пустыни 3
Бриллиантовая рука 2
Гараж 1
```

37) По каналу связи передается последовательность положительных целых чисел, все числа не превышают 1000, их количество заранее неизвестно. Каждое число передается отдельно. Признаком конца передаваемой последовательности является число 0. После числа 0 передается контрольное значение – наибольшее число R, удовлетворяющее следующим условиям:

1) R – произведение двух различных переданных элементов последовательности («различные» означает, что не рассматриваются квадраты переданных чисел, произведения различных, но равных по величине элементов допускаются);
2) R делится на 6

Напишите эффективную программу, которая получает последовательность чисел и следующие за ней признак конца и контрольное значение, а также проверяет правильность контрольного значения. Программа должна напечатать отчет по следующей форме:

```
Получено .. чисел
Полученное контрольное значение: ….
Вычисленное контрольное значение:…
Контроль пройден (или – контроль не пройден)
```

Размер памяти, которую использует Ваша программа, не должен зависеть от длины переданной последовательности чисел. Перед текстом программы кратко опишите используемый вами алгоритм решения задачи.

Пример входных данных:

```
60
17
3
7
9
60
```

```
0
3600
```

Пример выходных данных для приведенного выше примера входных данных:

```
Получено 6 чисел
Полученное контрольное значение: 3600
Вычисленное контрольное значение: 3600
Контроль пройден.
```