**Примерные задания для проведения школьного этапа олимпиады по программированию  
 в 9 – 11 классах 2015 – 2016 учебного года**

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, 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 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**

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

x2 обозначается как «a»

x обозначается как «b»

Например, уравнение  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**

1. Взаимным индексом совпадения строк *S­*1 и *S*­2, которые включают только латинские буквы, называется величина

,



где *n­*1 и *n*­2 – длины строк *S­*1 и *S*­2, а – число вхождений буквы, имеющей номер *k* в латинском алфавите, в строку *S­*i. Например, индекс совпадений строк «Moloko» и «mAma» равен



(одна общая буква «m» встречается 1 раз в первой строке и 2 раза во второй строке). Напишите эффективную программу, которая вводит с клавиатуры две строки, содержащие (кроме латинских букв) знаки препинания и пробелы, и вычисляет взаимный индекс совпадения этих строк.

1. Гоночная трасса состоит из двух основных дорог и нескольких переездов, позволяющих перейти с одной дороги на другую.

A0

B0

A1

A2

AN-1

AN

B1

B2

BN-1

BN

t

t

t

t

t

a1

a2

aN-1

bN-1

b2

b1

На всех участках, включая переезды, движение разрешено только в одну сторону, поэтому переезд возможен только с дороги A на дорогу B. Гонщик стартует в точке A0 и должен финишировать в точке BN. Он знает, за какое время сможет пройти каждый участок пути по каждой дороге, то есть время прохождения участков A0A1, A1A2, ..., AN-1AN, B0B1, B1B2, ..., BN-1BN. Время прохождения всех переездов A0B0, A1B1, ..., ANBN одинаково и известно гонщику. Необходимо определить, за какое минимальное время гонщик сможет пройти трассу.

Напишите эффективную, в том числе по используемой памяти, программу для решения этой задачи. Перед текстом программы кратко опишите алгоритм решения и укажите язык программирования и его версию.

**Входные данные**

В первой строке задаётся количество участков трассы N. Во второй строке задаётся целое число t – время (в секундах) прохождения каждого из переездов A0B0, A1B1, …, ANBN. В каждой из последующих N строк записано два целых числа ai и bi, задающих время (в секундах) прохождения очередного участка на каждой из дорог. В первой из этих строк указывается время прохождения участков A0A1 и B0B1, во второй – A1A2 и B1B2 и т. д.

**Пример входных данных**

**3**

**20**

**320 150**

**200 440**

**300 210**

**Выходные данные**

Программа должна напечатать одно целое число: минимально возможное

время прохождения трассы (в секундах).

**Пример выходных данных для приведённого выше примера входных данных**

**750**

**Рекомендации по проверке:**

1. В этой задаче не нужно хранить в памяти все отсчеты, нас интересуют только средние значения температуры по каждому месяцу и по году, поэтому алгоритм на Паскале таков:

**const DAYS = 366;**

**var tMonth: array[1..12] of real;**

**i, month: integer;**

**t, tYear, min: real;**

**c: char;**

**begin**

**for i:=1 to 12 do tMonth[i]:= 0;**

**tYear := 0;**

**for i:=1 to DAYS do begin**

**repeat read(c); until c = '.';**

**read (month);**

**readln (t);**

**tMonth[month] := tMonth[month] + t;**

**tYear := tYear + t;**

**end;**

**for i:=1 to 12 do**

**case i of**

**2: tMonth[i] := tMonth[i] / 29;**

**4,6,9,11: tMonth[i] := tMonth[i] / 30;**

**else tMonth[i] := tMonth[i] / 31;**

**end;**

**tYear := tYear / DAYS;**

**min := abs(tMonth[1] - tYear);**

**for i:=2 to 12 do**

**if abs(tMonth[i] - tYear) < min then**

**min := abs(tMonth[i] - tYear);**

**writeln(tYear:0:2);**

**for i:=1 to 12 do**

**if abs(tMonth[i] - tYear) = min then**

**writeln(i,' ',tMonth[i]:0:2,' ',**

**tMonth[i]-tYear:0:2);**

**end.**

1. В этой задаче нужно подсчитать, сколько раз встречается каждая буква. Если из букв можно составить палиндром, то одна буква (центральная) может встречаться нечетное число раз, а остальные – обязательно четное.

**var count: array['A'..'Z'] of integer;**

**i, nOdd: integer;**

**c, cOdd: char;**

**begin**

**for c:='A' to 'Z' do count[c] := 0;**

**read(c);**

**while c <> '.' do begin**

**count[c]:= count[c] + 1;**

**read(c);**

**end;**

**nOdd := 0;**

**for c:='A' to 'Z' do**

**if count[c] mod 2 = 1 then begin**

**cOdd := c;**

**Inc(nOdd);**

**end;**

**if nOdd > 1 then**

**writeln('Нет')**

**else begin**

**writeln('Да');**

**for c:='A' to 'Z' do**

**for i:=1 to count[c] div 2 do**

**write(c);**

**if nOdd = 1 then write(cOdd);**

**for c:='Z' downto 'A' do**

**for i:=1 to count[c] div 2 do**

**write(c);**

**end;**

**end.**

1. Особенность этой задачи состоит в том, что здесь жестко прописано, что вводить данные нужно из файла, а выводить результаты – в тот же файл, дописывая их в конец файла. Тут помогут стандартные функции для работы с файлами: **Assign** (связать указатель на файл с файлом на диске), **Reset** (открыть файл на чтение), **Rewrite** (открыть файл на запись), **Append** (открыть файл на добавление в конец) и **Close** (закрыть файл). Схема работы с файлами в данной программе выглядит так (считаем, что файл называется **eq.txt**):

**var F: Text; { указатель на текстовый файл }**

**s: string;**

**a, b, c: real; { коэффициенты уравнения }**

**D: real; { дискриминант }**

**x1, x2: real; { корни уравнения }**

**i, r: integer; { вспомогательные переменные }**

**begin**

**Assign(F, 'eq.txt');**

**Reset(F);**

**readln(F, s); { читаем первую строку файла }**

**Close(F);**

**{ обработка данных, запись корней уравнения в x1 и x2 }**

**Append(F);**

**writeln(F, x1); { вывод результатов в конец файла }**

**writeln(F, x2);**

**Close(F);**

**end.**

Обратите внимание на три момента:

* после завершения файловой операции (чтения, записи или добавления) файл нужно закрывать вызовом **Close**;
* после вызова **Close** мы можем снова использовать тот же указатель **F**;
* вызывать второй раз **Assign** не нужно, поскольку мы записываем в тот же файл.

Вторая проблема, которую часто не замечают, считая эту задачу слишком легкой. В условии не сказано, что коэффициенты уравнения перечисляются от старшего к младшему. То есть, строго говоря, уравнение  может быть задано 6 способами:

|  |  |  |
| --- | --- | --- |
| **2a-4b-6**  **2a-6-4b** | **-4b+2a-6**  **-4b-6+2a** | **-6+2a-4b**  **-6-4b+2a** |

Поэтому примитивный алгоритм (найти символ «a», слева от него – старший коэффициент и т.д.) не срабатывает. Тогда как искать коэффициенты?

Понятно, что нужно проходить вдоль строки, начиная с первого символа, и выделять очередной коэффициент. Рассмотрим алгоритм выделения первого (по порядку) коэффициента из строки s. Число заканчивается там, где встречается один из стоп-символов: '**a'**, '**b'**, '**+'** или '**-'**. Для удобства добавим в конец строки символ '!' – он нужен для того, чтобы остановиться в том случае, когда последний коэффициент – это свободный член уравнения, и за ним строка кончается.

**var F: Text;**

**s: string;**

**i, r: integer;**

**a, b, c, D, x1, x2: real;**

**begin**

**Assign(F, 'eq.txt');**

**Reset(F);**

**Readln(F, s);**

**Close(F);**

**s:=s+'!';**

**i:=1;**

**while i <= Length(s) do begin**

**case s[i] of**

**'a': begin**

**if i = 1 then a := 1 { нет коэффициента => a=1}**

**else**

**if (i = 2) and (s[1] in ['+','-'] then**

**if s[1] = '+' then a := 1 { a=1 или a=-1}**

**else a := -1**

**else**

**Val(Copy(s,1,i-1), a, r); { есть коэффициент }**

**s:=Copy(s, i+1, Length(s)-i);**

**i := 0;**

**end;**

**'b': begin**

**if i = 1 then b := 1 { нет коэффициента => b=1}**

**else**

**if (i = 2) and (s[1] in ['+','-'] then**

**if s[1] = '+' then b := 1 { b=1 или b=-1}**

**else b := -1**

**else**

**Val(Copy(s,1,i-1), b, r); { есть коэффициент }**

**s:=Copy(s, i+1, Length(s)-i);**

**i := 0;**

**end;**

**'+','-':**

**if i > 1 then begin**

**Val(Copy(s,1,i-1), c, r);**

**s:=Copy(s, i, Length(s)-i);**

**i := 0;**

**end;**

**'!':**

**if i > 1 then**

**Val(Copy(s,1,i-1), c, r);**

**end;**

**i:=i+1;**

**end;**

**D:= b\*b-4\*a\*c;**

**x1 := (-b - sqrt(D)) /(2\*a);**

**x2 := (-b + sqrt(D)) /(2\*a);**

**Append(F);**

**writeln(F, x1);**

**writeln(F, x2);**

**Close(F);**

1. С первого взгляда эта задача кажется достаточно простой: нужно ввести две строки, подсчитать количество латинских букв в каждой из них (записав эти данные в два массива счетчиков **f1** и **f2**), затем найти сумму произведений элементов этих массивов и разделить её на произведение длин строк.

В языке Паскаль индексами элементов массива могут быть символы, поэтому можно объявить массивы счетчиков и переменную **k** так:

**var k: char;**

**f1, f2: array['A'..'Z'] of integer;**

**Полная программа может выглядеть так:**

**var i, count1, count2, sumFF: integer;**

**s1, s2: string;**

**k: char;**

**begin**

**readln(s1);**

**readln(s2);**

**sumFF := 0;**

**for k:='A' to 'Z' do begin**

**count1 := 0;**

**for i:=1 to length(s1) do**

**if k = UpCase(s1[i]) then**

**count1 := count1 + 1;**

**count2 := 0;**

**for i:=1 to length(s2) do**

**if k = UpCase(s2[i]) then**

**count2 := count2 + 1;**

**sumFF := sumFF + count1 \* count2;**

**end;**

**if length(s1)\*length(s2) > 0 then**

**writeln(sumFF/(length(s1)\*length(s2)):10:3)**

**else**

**writeln(0);**

**end.**

1. Для решения этой задачи можно, например, определить, когда нужно перехать с дороги А на дорогу В так, чтобы общее время было минимальным. Пусть мы переехали по переезду AiBi. Тогда время общее время движения равно

**Ti = SAi + t + SBi**

Здесь **SAi** – это время проезда от **A0** до **A­i**;

t – время переезда между **Ai** до **Bi**;

**SBi** – это время проезда от **Bi** до **B­N**.

Из всех возможных значений **Ti** нужно выбрать минимальное, это и есть ответ.

Напишем сначала именно такое (неэффективное) решение. Для того, чтобы построить массив SA, будем накапливать частичные суммы:

**SAi = 0**

**SA1 = A1 = SA0 + A1**

**SA2 = A1 + A2 = SA1 + A2**

**...**

Все это укладывается в формулу **SAi = SAi-1 + Ai**, которую будем использовать в цикле для заполнения массива SA. Полная программа примет вид:

**var N, i, t, TA, TB, ai, bi: integer;**

**begin**

**Readln( N );**

**Readln( t );**

**TA := 0;**

**TB := t;**

**for i:=1 to N do begin**

**Readln( ai, bi );**

**TA := TA + ai;**

**if TB+bi < TA+t then**

**TB:= TB + bi**

**else TB:= TA + t;**

**end;**

**writeln(TB);**

**end.**