2016 год

Всероссийская олимпиада школьников по информатике
Муниципальный этап
9-11 классы

Методические рекомендации по разбору предложенных олимпиадных задач

Задача A. От A до B — 20 баллов

Ограничение по времени, сек.	1
Ограничение по памяти, мегабайт	1
Имя входного файла	`dist.in`
Имя выходного файла	`dist.out`

Автоматический поезд, следуя от станции A до станции B, должен посетить ряд промежуточных полустанков. Для этого он совершает два вида перемещений: проехать 2 километра в сторону станции B или вернуться на 3 километра в сторону станции A. Известно, что начиная своё движение на станции A, он заканчивает его на станции B.

По предложенному расписанию движения поезда определите расстояние от станции A до станции B.

Формат входного файла
Одна строка, состоящая из N ($1 \le N \le 10^3$) цифр 2 и 3, где 2 – перемещение на 2 километра в сторону станции B, 3 – перемещение на 3 километра обратно, в сторону станции A.

Формат выходного файла
Выходной файл должен содержать одно число S – расстояние от станции A до станции B.

Пример

dist.in	dist.out
22232	5

Решение
Очевидны несколько путей решения задачи. Во-первых, моделирование описанного в задаче процесса. Во-вторых, подсчёт количество цифр 2 и 3 отдельно в строке.

На языке Pascal решение может принять следующий вид:

```
BEGIN
//Чтение исходных данных
 var f: text := OpenRead('dist.in');
 var s: string;
 readln(f,s); CloseFile(f);
 var n: integer := 0;
//Моделирование описанного в задаче процесса
 for var i: integer := 1 to length(s) do
```

```
  if s[i] = '2' then n += 2 else n -= 3;
//Вывод результата
 f := OpenWrite('dist.out');
 write(f,n); CloseFile(f);
END.
```

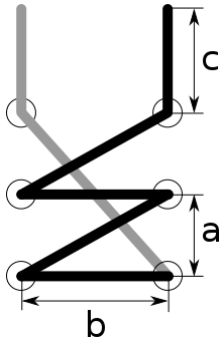
На языке Python решение может принять следующий вид:

```
# Чтение исходных данных
f = open('dist.in'); a = f.read().strip(); f.close()
# Моделирование описанного в задаче процесса
s = 0
for i in a:
    if i =='2':
        s +=2
    else: s -= 3
# Вывод результата
f = open('dist.out','w'); f.write(str(s)); f.close()
```

Задача B. Модная шнуровка — 20 баллов

Ограничение по времени, сек.	1
Ограничение по памяти, мегабайт	1
Имя входного файла	`bootlace.in`
Имя выходного файла	`bootlace.out`

Для завязывания шнурков в ботинках используется очень модная шнуровка, которая осуществляется с соблюдением принципов, изображённых на рисунке. Одна часть шнурка соединяет самое верхнее левое и нижнее правое отверстие для шнуровки, в то время как вторая часть шнурка, последовательно соединяет все отверстия. После осуществления шнуровки по указанным правилам должны остаться одинаковые остатки шнурка достаточные для завязывания очень модного банта.



Требуется по известным параметрам отверстий для шнурка и банта определить минимальную длину необходимого шнурка.

Формат входного файла

Одна строка, состоящая из четырёх натуральных чисел: нечётное N ($3 \le N \le 1000$), a, b и c ($1 \le a, b, c \le 100$). Где N – количество отверстий для шнуровки в одном вертикальном ряду, a – вертикальное расстояние между отверстиями, b – расстояние между рядами отверстий, c – длина остатка шнурка для завязывания модного банта.

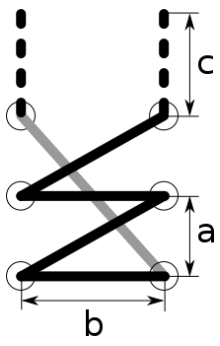
Формат выходного файла

Выходной файл должен содержать одно число – минимальная длина шнурка с точностью до сотых.

Пример

bootlace.in	bootlace.out
3 2 3 4	26.21

Решение



Оценим длину отдельных участков. Длина пунктирной линии (остатки шнурков, завязываемые бантом) $S_1 = 2 \cdot c$. Длина серой линии (внутренний шнурок, соединяющий верхнее-левое и нижнее-правое отверстие) $S_2 = \sqrt{b^2 + \left(a \cdot (n-1)\right)^2}$. Длина чёрной линии (основная шнуровка) $S_3 = \left(b + \sqrt{b^2 + a^2}\right) \cdot (n-1)$. Общая длина найдётся как $l = S_1 + S_2 + S_3$

На языке Pascal решение может принять следующий вид:

```
var n,a,b,c: integer; l: real;
BEGIN
//Чтение исходных данных
 var f: text := OpenRead('bootlace.in');
 read(f,n,a,b,c); CLoseFile(f);
//Расчёт длины шнуровки
 l := 2*c +
      sqrt(sqr(b)+sqr(a*(n-1))) +
      (b+sqrt(sqr(b)+sqr(a)))*(n-1);
//Вывод результата
 f := OpenWrite('bootlace.out');
 write(f,l:1:2); CLoseFile(f);
END.
```

На языке Python решение может принять следующий вид:

```
# Чтение исходных данных
f = open('bootlace.in')
n, a, b, c = map(int,f.read().strip().split())
f.close()
# Расчёт длины шнуровки
s = 2*c+n*b+(n-1)*(a*a+b*b)**0.5+(b*b+((n-1)*a)**2)**0.5-b
# Вывод результата
f = open('bootlace.out','w')
f.write('{:2.2f}'.format(s)); f.close()
```

Задача C. Мобильный оператор — 20 баллов

Ограничение по времени, сек.	1
Ограничение по памяти, мегабайт	1
Имя входного файла	`mobile.in`
Имя выходного файла	`mobile.out`

Новый сотовый оператор предлагает выгодные тарифы на голосовую связь и текстовые сообщения с посекундной тарификацией. При этом чтобы привлечь клиентов, каждый месяц он дарит пакет не тарифицируемых минут и сообщений, взимая при этом плату за все разговоры и сообщения сверх подарка. Оператор предоставляет ежемесячный отчёт о разговорах и

сообщениях, чтобы пользователь мог легко убедиться в честности оператора и правильности выставленного счёта.

Требуется помочь пользователю рассчитать требуемую для оплаты сумму.

Формат входного файла

В первой строке одно число N ($1 \le N \le 10^3$) – число записей о звонках и сообщениях. Вторая строка содержит два целых P_1 и P_2 и два вещественных S_1 и S_2 числа, разделённых пробелами ($1 \le P_1, P_2, S_1, S_2 \le 10^4$). P_1 и P_2 – не тарифицируемое (подарочное) количество минут и SMS соответственно. S_1 и S_2 – стоимость минуты и стоимость SMS соответственно.

Далее N строк – информация о звонках и сообщениях. Для звонка: дата, время, длительность звонка в формате MM:SS. Для сообщения: дата, время отсылки сообщения и кодовый текст «SMS». Формат даты DD.MM.YYYY, формат времени HH:MM:SS.

Формат выходного файла

Выходной файл должен содержать одно число – необходимая сумма оплаты с точностью до копейки.

Пример

mobil.in	mobil.out
7 10 3 1.20 0.53 01.11.2017 09:10:01 SMS 01.11.2017 10:03:57 SMS 02.11.2017 11:14:03 02:10 05.11.2017 13:12:56 SMS 06.11.2017 08:12:12 03:50 06.11.2017 17:55:21 04:34 30.11.2017 12:19:38 SMS	1.21

Решение

Основой при решении задачи является аккуратность в обработке предлагаемых данных. Обратим внимание на некоторые моменты. Во-первых, строк с информацией о звонке и о SMS имеют различие в длине. Во-вторых, оперировать с информацией о длительности звонков удобнее в секундах.

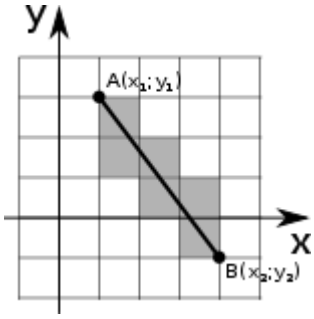
На языке Pascal решение может принять следующий вид:

```
var p1,p2,n,nsms,nring: integer; s1,s2,pr: real;
     s: string;
BEGIN
//Чтение исходных данных
 var f: text := OpenRead('mobil.in');
 readln(f,n); readln(f,p1,p2,s1,s2);
//Инициализация счётчиков количества SMS, длительности звонков в
//секундах
 nsms := 0; nring := 0;
```

```
//Чтение данных о звонках и SMS
 for var i: integer := 1 to n do begin
  readln(f,s);
  s := copy(s,21,length(s)-20);
  if length(s)=3 then nsms += 1
    else
     nring += StrToInt(copy(s,1,2))*60 + StrToInt(copy(s,4,2))
 end;
 CloseFile(f);
//Расчёт оплаты с учётом подаренных пакетов
 if nsms > p2 then pr := (nsms-p2) * s2;
 if nring/60 > p1 then pr += (nring-p1*60) * s1/60;
//Вывод результата
 f := OpenWrite('mobil.out');
 writeln(f, pr:1:2); CloseFile(f);
END.
```

Задача D. Путешествие — 20 баллов

Ограничение по времени, сек.	2
Ограничение по памяти, мегабайт	1
Имя входного файла	`walk.in`
Имя выходного файла	`walk.out`



Квадратная страна поделена на правильные единичные квадратные секторы – области страны. Путник, двигаясь из точки A в точку B по прямолинейной траектории, посещает несколько областей. Если путник прошёл через границу области, то не считается, что он её посетил, посещение регистрируется только при посещении территории области.

Требуется по известным координатам начала путешествия A и его завершения B определить, сколько областей посетил путник.

Формат входного файла

В первой строке четыре целых числа x_1, y_1, x_2, y_2 ($-10^3 \le x_1, y_1, x_2, y_2 \le 10^3$), разделённых пробелом, – координаты точек начала $A(x_1; y_1)$ и конца $B(x_2; y_2)$ отрезка.

Формат выходного файла

Выходной файл должен содержать одно число – количество пересечённых клеток.

Пример

walk.in	walk.out
1 3 4 -1	6
2 0 2 5	0
-1 -1 4 4	5

Решение
Одним из возможных решений может быть определение закономерности в количестве пересечённых клеток в зависимости от координат отрезка, которая имеет следующий вид: $S = \Delta x + \Delta y - \text{НОД}(\Delta x \cdot \Delta y)$, где Δ – модуль разности соответствующих координат концов отрезка.

На языке Python решение может принять следующий вид:

```
# функция поиска НОД
def nod(a,b):
    while a != 0 and b != 0:
        if a > b: a %= b
        else: b %= a
    if a == 0: return b
    return a
# Чтение исходных данных
f = open('walk.in')
x1,y1,x2,y2 = map(int,f.readline().split())
f.close()
x = abs(x1-x2)
y = abs(y1-y2)
k = nod(x,y)
S = x+y-k
# Вывод результата
f = open('walk.out','w'); f.write(str(S)); f.close()
```

Задача E. Неклассические классики — 20 баллов

Ограничение по времени, сек.	1
Ограничение по памяти, мегабайт	1
Имя входного файла	`game.in`
Имя выходного файла	`game.out`

Дети играли в игру на асфальте. Полем для игры выступала дорожка, разделённая на квадраты. В каждом квадрате дорожки написано некоторое натуральное число. Игрок в начальный момент времени находится перед первым квадратом и может совершать следующие движения: прыгнуть на следующий квадрат, перепрыгнуть квадрат. Попадая на определённый квадрат дорожки, игрок прибавляет к своему счёту число, написанное в квадрате, на который он попал. До начала игры счёт игрока равен нулю. Задача игрока - «пройти» дорожку, набрав минимальное количество очков.

Требуется определить минимальную сумму баллов, за которую игрок может пройти дорожку.

Формат входного файла
В первой строке одно число N ($2 \le N \le 20$) – длина дорожки (количество квадратов). Во второй строке N натуральных чисел [1; 100] – последовательно значения, написанные на квадратах.

Формат выходного файла
Выходной файл должен содержать одно число – минимальная сумма баллов за которую можно пройти игру.

Пример

game.in	game.out
5 2 1 1 5 1	3

Решение
Один из вариантов сводится к перебору возможных путей прохода по игровой дорожке. При этом сумма баллов прохождения по этому пути становится предметом для выявления минимального путём сравнения с предыдущим значением.
В предложенном варианте решения перебираются все теоретически возможные пути (функция NextPath), а каждый путь оценивается на соответствие условию задачи (функция Test).

На языке Pascal решение может принять следующий вид:

```
var d: array of integer; n: integer;
//Функция позволяет сгенерировать следующий маршрут движения по
//полю: 1 - клетка на которую прыгаем, 0 - клетка через которую
//прыгаем.
function NextPath(a: string): string;
begin
 for var i:integer := length(a) downto 1 do
  if a[i] = '1' then a[i] := '0' else begin
   a[i] := '1'; break;
  end;
 result := a;
end;
//Функция проверки на соответствие маршрута, описанным в задаче
//граничным условиям
function Test(s: string): boolean;
begin
 for var i: integer := 1 to length(s)-1 do
  if (s[i]='0')and(s[i+1]='0') then
    begin Result := false; exit; end;
 Result := true;
end;
//Функция определения суммы значений при прохождении
//по пути (аргумент)
function SumPath(s: string): integer;
begin
  var sm: integer := 0;
  for i: integer := 1 to length(s) do
   if s[i] = '1' then sm += d[i-1];
  result := sm;
end;

 var sm,rmin: longint;
```

```
BEGIN
 var f: text := OpenRead('game.in');
//Чтение данных из файла в массив d
 readln(f,n); SetLength(d,n);
 for var i: integer := 0 to n-1 do read(f,d[i]);
 CloseFile(f);
//Определяемся с начальным значением минимального rmin
 if n = 2 then rmin := min(d[0],d[1]) else rmin := 2000;
//Начинаем подготовку к перебору возможных путей
 if n > 2 then begin
   var st: string := ''; var stfin: string := '';
   for var i: integer := 1 to n-2 do st := st+'0';
   stfin := '00'+st; st := '01'+st;
//Начинаем перебор возможных путей
   Repeat
//Если путь соответствует условиям задачи, то ищем сумму
    if Test(st) then begin
      sm := SumPath(st);
//Выбираем минимальную сумму
      if sm < rmin then rmin := sm;
    end;
    st := NextPath(st)
   until st = stfin;
 end;
//Вывод результата
 f := OpenWrite('game.out'); write(f,rmin); CloseFile(f);
END.
```