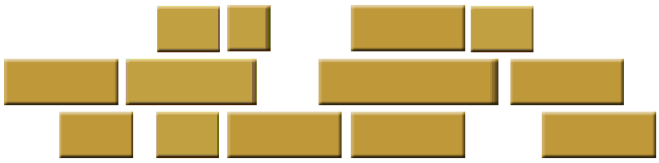
Оглавление

7-8 класс

Задача 1. Крепостная стена (100 баллов)

Имя входного файла:	fortress.in
Имя выходного файла:	fortress.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб



Имеется древняя крепостная стена длиной L из N прямоугольных камней разного размера. Никакой камень не вылезает за границы стены. Стена очень древняя и часть камней разрушилась и выпала, образовав сквозные пустоты. Археологам необходимо замерить количество камней под заданными M точками стены. Точки задаются натуральным числом – расстоянием от левого края стены.

От Вас **требуется** написать программу расчета количества камней под заданными точками по заданным координатам всех камней в стене.

Формат входного файла

В первой строке входного файла записаны 3 целых числа:

- натуральное число L – длина стены ($1 \le L \le 10\,000$);
- натуральное число N – количество камней в стене ($1 \le N < 10\,000$);
- натуральное число M – число точек стены, под которыми необходимо определить количество камней ($1 \le K < 100\,000$).

В следующих N строках идут пары натуральных чисел l и r ($1 \le l \le r \le L$) – левые и правые концы камней стены.

В следующих M строках идут натуральные числа q ($1 \le q \le L$) – координаты точек стены, под которыми нужно вычислить количество камней

Формат выходного файла

В M строках выходного файла вывести по одному целому числу – количество камней над очередной точкой.

Пример входного и выходного файлов

fortress.in	fortress.out
39 4 7 3 21 3 15 2 20 3 17 4 17 33	4 3 0 4 4 0 0

5 9 25 37	

Рисунок, поясняющий пример.

Решение

Задача относится к теме «сортировка подсчетом». Решение сводится к нахождению количества кирпичей под каждой точкой стены, имеющей целочисленные координаты. Для этого заводим массив или список и в ходе ввода координат кирпичей точки, на которые попадает этот кирпич, увеличиваем на 1. Затем при вводе координат точек сразу выводим значение соответствующего элемента.

Решение на PascalABC:

```
Var l,j,n,m,i,k,t,h:integer;
a:array[1..10000]of integer;
begin
  assign(input,'fortress.in');
  reset(input);
  readln(l,n,m);
  for i:=1 to n do
  begin
    readln(t,h);
    for j:=t to h do a[j]+=1;
  end;
  assign(output,'fortress.out');
  rewrite(output);
  for i:=1 to m do
  begin
    readln(t);
    writeln(a[t]);
  end;
  close(input);
  close(output);
end.
```

Решение на Python:

```
l, n, m = map(int, input().split())
a = [0 for i in range(l+1)]
for i in range (1,n+1):
```

```
    t, h = map(int, input().split())
    for j in range (t, h+1):
        a [j] += 1
for i in range(1,m+1):
    t = int(input())
    print(a[t])
```

Задача 2. Красивые обои (100 баллов)

Имя входного файла:	wallpaper.in
Имя выходного файла:	wallpaper.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб

Петя - будущий программист. А его старший брат Володя – дизайнер. Он создает новые рисунки на выпускающем обои предприятии. Заказчик попросил создать различным образом разрисованные прямоугольниками обои. Заказ был мало творческим, скорее механическим, поэтому Володя попросил Петю попробовать написать программу, которая будет рисовать рамки прямоугольников.

Поле для будущей композиции – тоже прямоугольник, который состоит из строк (пронумерованных от 1 до X) и столбцов (пронумерованных от 1 до Y) Нумерация соответственно сверху-вниз и слева-направо. На поле необходимо вывести N рамок прямоугольников, заданных двумя точками ($w1_i$, $h1_i$) и ($w2_i$, $h2_i$) - координаты противоположных углов, лежащих на диагонали i-того прямоугольника.

Для оценки гармонии рисунка программа должна создавать изображение с помощью букв латинского алфавита, i-ый прямоугольник создается i-ой буквой алфавита, а свободные места заполнять символом 0. Прямоугольник, нарисованный позже, может перекрывать ранее нарисованные.

Итак, от Вас **требуется** написать программу, которая будет рисовать рамки прямоугольников на поле обоев.

Формат входных данных

Первая строка содержит 3 натуральных числа X, Y и N – размеры композиции и число прямоугольников ($2 \le X \le 100$, $2 \le Y \le 100$, $1 \le N \le 26$).

Следующие N строк содержащих координаты прямоугольников – натуральные числа $w1_i$, $h1_i$, $w2_i$, $h2_i$ ($1 \le w1_i < w2_i \le X$, $1 \le h1_i < h2_i \le Y$).

Формат выходных данных

X строк композиции, заполненных в соответствии со входными данными

Пример входного и выходного файлов

wallpaper.in	wallpaper.out
10 10 3 2 4 9 7 6 6 10 10 1 1 3 8	ccccccc00 c00aaaac00 ccccccc00 000a00a000 000a00a000 000a0bbbbb 000a0ba00b 000a0ba00b 000aaba00b 00000bbbbb

Решение

Задача на работу со стандартными структурами данных. Сделаем в точности то, что просят в условии. Заведем двумерный массив размером x на y. Для каждого прямоугольника заполним его рамку соответствующими символами. Выводим заполненный массив по строкам. Приведем пример решения задачи на различных языках программирования.

Решение на PascalABC:

```
program Wallpaper;
var
  i, j, k, a, b, c, d, h, w, n: integer;
  ans: array[1..80, 1..80] of char;
  cur: char;
begin
  assign(input,'wallpaper.in');
  reset(input);
  readln(h, w, n);
  for i := 1 to h do
    for j := 1 to w do ans[i, j] := '0';
  for k := 0 to n - 1 do
  begin
    cur := Chr(Ord('a') + k);
    read(a, b, c, d);
    for i := a to c do
    begin
      ans[i, b] := cur;
      ans[i, d] := cur
    end;
    for i := b to d do
    begin
      ans[a, i] := cur;
      ans[c, i] := cur
    end
  end;
  assign(output,'wallpaper.out');
  rewrite(output);
  for i := 1 to h do
  begin
    for j := 1 to w do write(ans[i, j]);
    writeln;
  end
  close(input);
  close(output);
end.
```

Решение на Python:

```
h, w, n = map(int, input().split())
c = [['0'] * w for i in range(h)]
for i in range(n):
    r1,c1,r2,c2=map(lambda x:int(x)-1,input().split())
    for p in range(c1, c2 + 1):
        c[r1][p] = chr(ord('a') + i)
        c[r2][p] = chr(ord('a') + i)
    for p in range(r1, r2 + 1):
        c[p][c1] = chr(ord('a') + i)
        c[p][c2] = chr(ord('a') + i)
for s in c:  print("".join(s))
```

Задача 3. Олимпиада (100 баллов)

Имя входного файла:	olympic.in
Имя выходного файла:	olympic.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб

Олимпиада по информатике в школах области проходила два дня, причем в первый день были предложены более простые для решения задачи, а во второй день задачи более высокого уровня сложности. Результаты олимпиады отсортировали по каждому дню в порядке убывания баллов, полученных каждым участником, и разместили в таблице так, что за самым низким баллом первого дня шел самый высокий балл второго дня. Для анализа результатов олимпиады потребовалось определить самое большое различие в баллах двух соседних результатов таблицы. Информация о результатах олимпиады содержала данные о результатах обоих дней, которые приходили постепенно.

От Вас **требуется** написать программу, определяющую указанное в условии различие.

Формат входных данных

Первая строка содержит целое число N – количество участников олимпиады ($2 \le N \le 200$).

Следующие N строк два числа d_i и b_i – день и балл i-го участника (d_i равно 1 или 2 и $0 \le b_i \le 100$)

Формат выходных данных

Одно число – максимальная разница в баллах стоящих рядом в отсортированной по дням таблице.

Пример входного и выходного файлов

olympic.in	olympic.out
8 1 23 2 38 1 33 1 85 2 66 1 22 2 44 2 52	52

Решение

Задача на использование стандартных алгоритмов обработки структур данных – алгоритмов сортировки. Отсортируем данные в соответствии с принципом, описанным в условии. После этого осталось найти максимум разностей соседних элементов. Сделаем это линейным проходом по массиву. Поскольку ограничения на число участников олимпиады были достаточно маленькие, можно воспользоваться любым алгоритмом сортировки, например, сортировкой пузырьком.

Решение на PascalABC:

```
program Olympic;
var
  i, j, a, h, n, bc, gc, tmp, ans: integer;
  b, g: array[1..50] of integer;
begin
  assign(input,'olympic.in');
  reset(input);
  readln(n);
  bc := 0;   gc := 0;
  for i := 1 to n do
  begin
    readln(a, h);
    if a = 0
    then begin inc(bc); b[bc]:=h end
    else begin inc(gc); g[gc]:=h end
  end;
  for i := 1 to bc do
    for j := i + 1 to bc do
      if b[i] < b[j]
      then begin tmp:=b[i]; b[i]:=b[j]; b[j]:=tmp end;
      for i := 1 to gc do
        for j := i + 1 to gc do
        if g[i] < g[j]
        then begin tmp:=g[i];g[i]:=g[j];g[j]:=tmp end;
  ans := 0;
  for i := 1 to bc - 1 do
    if ans < b[i]-b[i+1] then ans:= b[i]-b[i+1];
  for i := 1 to gc - 1 do
    if ans < g[i]-g[i+1] then ans:= g[i]-g[i+1];
  if (bc>0) and (gc>0)
  then if ans < abs(g[1] - b[bc])
       then ans:=abs(g[1]-b[bc]);
  assign(output,'olympic.out');
  rewrite(output);
  writeln(ans);
  close(input);
  close(output);
end.
```

Решение на Python, использующее встроенную сортировку:

```
n = int(input())
p = []
for i in range(n):
     a, h = map(int, input().split())
     p.append((a, h))
```

```
        p.sort ( key = lambda v: (v[0], -v[1]) )
        print (p)
ans = max([p[i-1][1] - p[i][1] for i in range(1, n)])
print(ans)
```

Задача 4. Волшебный конь (100 баллов)

Имя входного файла:	horse.in
Имя выходного файла:	horse.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб

В волшебной стране Дивнопутии жил волшебный конь Бегень. Особенность Бегня заключалась в том, что после долгого застоя, когда конь никуда не скакал, у него происходил упадок сил, и поэтому, отправляясь в очередной путь, Бегень мог не преодолеть весь путь без остановки и подзаправки волшебным сеном и родниковой водой. Но каждая пробежка увеличивала силы коня и после небольшого подкрепления каждый следующий участок пути, который преодолевал Бегень, был больше предыдущего, по крайней мере, на 2 км. Известно также, что каждый участок пути, преодолённый конем, это целое число километров.

Итак, от Вас **требуется** написать программу, которая выведет все возможные для заданного расстояния разбиения пути Бегня на отрезки, соответствующие условию задачи.

Формат входных данных

Входной файл содержит одно натуральное число N – количество километров, которое должен проскакать Бегень $(1 \le N \le 150)$.

Формат выходных данных

Выходной файл содержит одно натуральное число K – количество разбиений заданного расстояния на отрезки пути коня.

Пример входного и выходного файлов

horse.in	horse.out
10	6

Пояснения к полученному результату – разбиения расстояния 10 км на отрезки пути коня:

10=1+3+6
10=1+9
10=2+8
10=3+7
10=4+6
10=10

Решение

Задача относится к теме «динамическое программирование». Для подсчета всех искомых разбиений пути коня применим рекурсию. В качестве параметров будем передавать уже сгенерированный фрагмент разбиения, последнее использованное в нем число и расстояние, которое осталось набрать. Когда разбиение полностью готово, считаем его.

Решение на PascalABC, генерирующее (но не выводящее!) разбиение в массиве:

```
program horse;
var   i, p, n, k: integer;
```

```
  a: array[1..80] of integer;
procedure gen(cur, len, n: integer);
var   i: integer;
begin
  if (cur = 0) then
    if len > 1
    then k +=1;
    else
     if cur > 0
     then
     begin
       p := -1;
       if len>1 then  p:=a[len-1];
       for i := p + 2 to cur do
       begin a[len]:= i; gen(cur - i, len + 1, n) end
     end
end;

begin
  res:=0;
  assign(input,'horse.in');
  reset(input);
  read(n);
  gen(n, 1, n);
  assign(output, 'horse.out');
  rewrite(output);
  write(k);
  close(input);
  close(output);
end.
```

Решение на Python, использующее тот же подход:

```
k = 0

def gen(s, n, x, a):
    global k
    if x == 0:
        k += 1
        return
    for b in range(x, a - 1, -1):
        if x - b >= b + 2 or x == b:
            gen(s + [b], n, x - b, b + 2)

n = int(input())
gen([], n, n, 1)
print (k)
```