## Готуємось до олімпіади 2015-1

## 1. Одна чверть

## Дві точки на площині, що не лежать на координатних осях, задані своїми координатами: А(Х1, Y1) і B(X2, Y2). Перевірити, чи лежать ці точки в одній координатній чверті.

|  |
| --- |
| Ідея розв'язання суто математична, вона цілком прозора з приведеного фрагменту:  ***...***  ***if (x1\*x2 > 0) and (y1\*y2 > 0) then writeln (' Одна чверть ')***  ***else writeln (' Різні чверті ');*** |

## 2. Кількість днів між датами (<http://www.e-olymp.com/uk/problems/147>)

Дано дві календарні дати. Визначити кількість днів між цими датами. Не забудьте, що високосним вважається рік, номер якого кратний чотирьом та не кратний 100, або кратний 400.

## В першому і другому рядку вхідного файлу записано по одній календарній даті у форматі D M Y ( день місяць рік). 1 ≤ D ≤ 31, 1 ≤ M ≤ 12,1 ≤ Y ≤ 2100. У вихідний файл потрібно записати одне число – кількість днів між датами

Наприклад,

1 12 2008 31 12 2008 31

|  |
| --- |
| Один з способів розв'язання описано при розгляді задачі Стаж (Житомирська обласна олімпіада 1988 року). Розглянемо ще один спосіб, описаний в бібліотеках програм для програмованих калькуляторів [4].  При виконанні фінансово-економічних розрахунків часто виникає необхідність не тільки визначати кількість днів ***N***, що пройшли між двома датами, а й у визначені дня тижня по даті числа. При відліку дат, починаючи з 1582 року, ця задача розв'язується при допомозі такого алгоритму:   1. Вводимо число ***D***, місяць ***М*** і рік ***Y***. 2. Обчислюємо функціонал ***F*** для січня і лютого (***М*** = 1, 2) за формулою:   ***F*** = 365***Y*** + ***D*** + 31(***M***–1) + ***Int***((***Y***–1)/4) – ***Int***(3/4 ***Int***(((***Y***–1)/100)+1)),  а для інших місяців – за формулою:  ***F***=365***Y*** + ***D*** + 31(***M***–1) – ***Int***(0,4***М***+2,3) + ***Int***(Y/4) – ***Int***(3/4***Int***(***Y***/100)+1),   1. День тижня ***DТ*** обчислюється за формулою:   ***DТ*** = ***F*** + 7(–***Іnt***(***F***/7)),  причому ***DТ*** = 0 – субота, 1 – неділя, 2 – понеділок, ... 6 – п'ятниця.   1. Для знаходження ***N*** виконуємо пункти 1 – 3 для обох дат і обчислюємо ***N*** = ***Abs***(***F1***–***F2***), де ***Abs*** – модуль числа, ***F1*** – значення функціоналу для однієї дати, а ***F2*** – значення функціоналу для іншої дати.   Спробуйте самостійно програмно реалізувати описаний алгоритм для сформульованої задачі. |

## 3. Чотиризначні числа

Знайти всі чотиризначні числа, кожне з яких записано різними цифрами і має наступні властивості: якщо цифри шуканого числа деяким чином переставити місцями і одержане таким способом нове чотиризначне число відняти від шуканого, то різницею буде чотиризначне число, записане тими ж цифрами. Вказати кількість таких чисел.

|  |
| --- |
| Задачу можна розв'язувати різними способами. Ми для розв'язання задачі використаємо поняття множин. Крім того, оскільки з кожного чотиризначного числа можна утворити 24 різні числа, то для генерації всіх можливих чисел з цифр заданого числа використаємо лексикографічний спосіб генерації перестановок.  Весь хід розв'язання можна умовно розділити на три етапи:   1. Знаходження чотиризначного числа, утвореного чотирма різними цифрами. 2. Генерування всіх можливих чотиризначних чисел, утворених цифрами заданого числа. 3. Перевірка трьох чисел з множини утворених чотиризначних чисел на відповідність умові задачі.   Всі етапи виділено коментарями у тексті програми, як пояснено і сам процес лексикографічного методу утворення перестановок, тому труднощів при аналізі програми не повинно бути. Але для кращого розуміння блоку генерації чотиризначних чисел лексикографічним методом коротко опишемо сам метод.  Будемо поки що розглядати не множину цифр числа, а множину порядкових номерів цифр у цьому числі. Оскільки число складається з чотирьох цифр, то ми маємо множину з чотирьох елементів: 1, 2, 3, 4. Нам потрібно знайти всі розміщення з чотирьох цифр по чотири цифри. У загальному випадку кількість розміщень з n елементів по m –знаходиться за формулою:    Але з точки зору інформатики нам зручніше використовувати ту ж формулу, але записану у вигляді: **=** *n ⋅ (n-1) ⋅ (n-2) ⋅ ... ⋅ (n-(m-2)) ⋅ (n-(m-1))*. Зручність використання полягає в тому, що у циклах нам зручніше працювати з цілочисельними змінними, а операція ділення вимагає використання дійсних типів. Використавши формулу, маємо: =4,= 12,= 24,= 24. Отже, нам потрібно утворити = 24 різних чотиризначних числа. Тобто, з точки зору інформатики задача поки що мало цікава, але ось вирішення питання виведення всіх варіантів новоутворених чисел у масив (або на екран) робить подальший розв’язок цієї задачі самостійним методом для розв’язання великої групи завдань. На даний момент нам потрібно навчитись утворювати ***n***–значні числа з набору ***n*** цифр (у термінах інформатики задачу можна сформулювати так: скласти програму для отримання всіх перестановок з ***n*** символів). Утворювати числа нам допоможуть порядкові номери цифр, з яких складається задане число. Оскільки множина порядкових номерів цифр числа на початку є впорядкованою за зростанням: 1, 2, 3, 4, то будемо діяти згідно алгоритму генерації перестановок лексикографічним методом наступним чином:   1. вивести на друк знайдене число; 2. починаючи з кінця слова, шукаємо перший елемент, який менший за попередній (рахуючи з кінця), і відмічаємо його; 3. знову починаємо з кінця і шукаємо ще один елемент, але вже більший за відмічений, причому він повинен бути розміщеним до відміченого, рахуючи з кінця; 4. якщо другий елемент знайдено, то міняємо його місцями з першим; 5. хвіст, що утворився після обміну (після номера елемента, знайденого першим), сортуємо в порядку зростання. Оскільки до цього він був впорядкований в порядку спадання, то у даному випадку операція сортування зводиться до обміну місцями відповідних елементів хвоста, або іншими словами – хвоста потрібно записати навпаки; 6. все вищесказане виконуємо до тих пір, доки весь масив не буде впорядковано в порядку спадання.   Все вищесказане і становить собою алгоритм генерації всіх перестановок натуральних чисел лексикографічним методом. У програмі будемо генерувати перестановки для натуральних чисел 1–4, що є номерами місцезнаходження цифр в утвореному чотиризначному числі, тобто індексами отриманого масиву цифр. Все інше легко зрозуміле з тексту програми.  ***program cislo4;***  ***type cis = 1000..9999; {*** діапазон чисел, що відповідають умові ***}***  ***cif = 0..9; {*** діапазон значень для цифр***}***  ***var kol : integer; {*** кількість знайдених пар чисел ***}***  ***i, j, n, k, t, m : byte;***  ***temp : cif;***  ***zero : integer;***  ***c : array[1..24] of cis; {*** для зберігання чотиризначних чисел ***}***  ***c1, c2, c3 : array[1..50] of cis; {*** для зберігання трійок чисел ***}***  ***a, b, num : array[1..4] of cif; {*** для генерації перестановок ***}***  ***flag, flag1, flag2 : boolean;***  ***begin***  ***kol := 0; {*** кількість трійок потрібних чисел = 0 ***}***  ***for b[1] := 1 to 9 do***  ***for b[2] := 0 to 9 do***  ***for b[3] := 0 to 9 do***  ***for b[4] := 0 to 9 do***  ***if (b[1]<>b[2]) and (b[1]<>b[3]) and (b[1]<>b[4]) and***  ***(b[2]<>b[3]) and (b[2]<>b[4]) and (b[3]<>b[4]) then***  ***begin***  ***a := b;***  ***{*** відсортували цифри a1 – a4 ***}***  ***for i := 1 to 3 do***  ***for j := i+1 to 4 do***  ***if a[i] > a[j] then***  ***begin***  ***temp := a[i]; a[i] := a[j]; a[j] := temp;***  ***end;***  ***{*** створення масиву ***с[1..24]*** в лексикографічному порядку ***}***  ***for i:=1 to 4 do num[i] := i; {*** заповнюємо масив номерами цифр ***}***  ***k := 1; {*** припускаємо, що існує хоча б одна перестановка ***}***  ***n:= 0; {*** множина шуканих чисел пуста ***}***  ***while k<>0 do {*** поки можлива хоча б одна перестановка ***}***  ***begin***  ***zero:=1000\*a[num[1]]+100\*a[num[2]]+10\*a[num[3]]+a[num[4]];***  ***if zero > 1000 then begin inc(n);c[n] := zero; end;***  ***j := 4; {*** починаючи з кінця масиву цифр ***}***  ***k := 0; {*** припускаємо, що перестановок більше немає ***}***  ***while (k=0) and (j>1) do {*** доки не знайшли чергову перестановку ***}***  ***begin {*** і не досягли початку масиву ***}***  ***if num[j-1] < num[j] then k := j -1; {*** знайшли першу цифру, ***}***  ***dec(j); {*** меншу за попередню, рахуючи з кінця ***}***  ***end; {*** якщо не знайшли, то вже всі перестановки знайдено ***(k=0) }***  ***j := 4; {*** переходимо до пошуку другої цифри ***}***  ***i := 0; {*** і припускаємо, що її знайти не можна ***}***  ***while (i=0) and (j>k) do***  ***{*** доки не визначили положення другої цифри і не досягли положення першої ***}***  ***begin {*** шукаємо другу цифру, яка більша першої відміченої ***}***  ***if num[j]>num[k] then i := j;***  ***dec(j); {*** причому також йдемо з кінця ***}***  ***end; {*** і до першої відміченої ***}***  ***if i>0 then {*** другу цифру знайдено – міняємо її місцями з першою ***}***  ***begin***  ***temp := num[i];***  ***num[i] := num[k];***  ***num[k] := temp;***  ***end;***  ***m := 1; {*** сортуємо "хвіст" в порядку зростання ***}***  ***while m < (4-k+1)/2 do***  ***begin***  ***temp := num[k + m];***  ***num[k + m] := num[5 - m];***  ***num[5 - m] := temp;***  ***inc(m);***  ***end;***  ***end;***  ***{*** кінець лексикографічного методу генерації чисел перестановкою цифр ***}***  ***{*** перевірка властивостей утворених чотиризначних чисел ***}***  ***for i := 1 to n do***  ***for j := 1 to n do***  ***begin***  ***zero := c[i] - c[j];***  ***if zero > 1000 then***  ***for t := 1 to n do***  ***if zero = c[t] then***  ***begin***  ***flag := true;***  ***for m :=1 to kol do if c2[m]=c[j] then flag:= false;***  ***flag1 := true;***  ***for m := 1 to kol do if c1[m]=c[i] then flag1:= false;***  ***flag2 := true;***  ***for m:=1 to kol do if c3[m]=zero then flag2:= false;***  ***if (flag = true) or (flag1 = true) or (flag2 = true) then***  ***begin***  ***inc(kol);***  ***c1[kol] := c[i]; c2[kol] := c[j]; c3[kol] := zero;***  ***write(c1[kol],' - ',c2[kol],' = ',c3[kol],' ');***  ***end;***  ***end;***  ***end;***  ***end;***  ***writeln;***  ***writeln(' Всього таких трійок чисел ', kol);***  ***end.*** |

## 4. Заміна нулів

Дано лінійну таблицю заданої довжини ***N***, яка містить велику кількість нульових елементів. Скласти на алгоритмічній мові алгоритм, що заміняє кожну групу нулів, що йдуть підряд на:

а) один нульовий елемент, якщо число таких нулів не парне;

б) два нульових елементи, якщо число таких нулів парне.

|  |
| --- |
| Здійснимо відповідні заміни за один прохід по масиву. Домовимось, що всі елементи в масиві типу integer і менші за найбільший елемент, який рівний ***maxint***. Ця домовленість потрібна для того, щоб хвіст вихідної лінійної таблиці заповнити максимальними елементами. Під час проходження по масиву потрібно постійно пам'ятати місце, куди переносимо відповідний елемент і номер елемента масиву, з якого починається хвіст. Коли хвіст стане рівним нулю, то, починаючи з поточного номера елемента до кінця заповнюємо масив значенням ***maxint***. Програмну реалізацію описаного вище алгоритму залишаємо за вами. |

## 5. Зафарбовані відрізки

На прямій зафарбували ***N*** відрізків. Відомі ***L***(***і***), ***R***(***і***) – ліві і праві кінці відрізків. Знайти суму довжин усіх зафарбованих частин прямої.

Потрібно відсортувати всі відрізки за неспаданням по лівих кінцях відрізків. Після цього порівнюємо кожен наступний з попереднім. Якщо його лівий кінець попадає між кінцями попереднього відрізка, то встановлюємо значення координати лівого кінця обох відрізків рівним значенню лівого кінця попереднього відрізка, а праві кінці обох відрізків набувають значення більшої з координат розглядуваних відрізків. При підрахунку суми ми збільшуємо її на різницю правих координат, якщо ліві координати відрізків рівні і на довжину розглядуваного відрізка в противному випадку. Описаний алгоритм повністю реалізовано в програмі.

***program line;***

***const m = 20;***

***var x : array[1..2,1..m] of integer;***

***i, j, p, n, s : integer;***

***begin***

***write('Кiлькiсть вiдрізкiв = '); readln(n);***

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

***begin***

***writeln('Введiть координати кiнцiв ', i ,'-го вiдрiзка: ');***

***write('Лiвий кiнець: X[1] = '); readln(X[1,i]);***

***write('Правий кiнець: X[2] = '); readln(X[2,i]);***

***end;***

***{*** сортування по лiвих кiнцях вiдрізкiв ***}***

***for i := 1 to n-1 do***

***for j := i+1 to n do***

***if x[1,j] < x[1,i] then***

***begin***

***p := x[1,i]; x[1,i] := x[1,j]; x[1,j] := p;***

***p := x[2,i]; x[2,i] := x[2,j]; x[2,j] := p;***

***end;***

***{*** алгоритм утворення нових кiнцiв правих i лiвих кiнцiв вiдрiзкiв ***}***

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

***if x[1,i] <= x[2,i-1] then***

***if x[1,i] >= x[1,i-1] then***

***begin***

***x[2,i-1] := x[2,i];***

***x[1,i] := x[1,i-1];***

***end***

***else begin***

***x[2,i] := x[2,i-1]; x[1,i] := x[1,i-1];***

***end;***

***{*** пiдрахунок сумарної довжини ***}***

***s := x[2,1] - x[1,1];***

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

***begin***

***if x[1,i] <> x[1,i-1] then s := s + (x[2,i] - x[1,i])***

***else s := s + x[2,i] - x[2,i-1];***

***end;***

***writeln(s);***

***end.***

## 6. Заповнення по діагоналях

Скласти алгоритм заповнення двомірної таблиці ***A***[1:***n***,1:**n**] по діагоналях з північного сходу на південний захід числами 1, 2, ..., ***n***2, починаючи з північно–західного кута таблиці.

|  |
| --- |
| Суто технічна задача, розрахована на перевірку вміння працювати з таблицями. Спосіб заповнення таблиці цілком зрозумілий з тексту програми.  ***program diagonal;***  ***const m = 100;***  ***var n, i, j, k, l, pi, pj : integer;***  ***a : array[1..m,1..m] of integer;***  ***begin***  ***write('N = '); readln(n);***  ***i := 1; j := 1; pi := 1; pj :=1 ; k := 1; l := n\*n;***  ***while k <= l do***  ***begin***  ***a[i,j] := k;***  ***inc(k);***  ***dec(j); inc(i);***  ***if (j < 1) or (i > n) then begin***  ***inc(pj);***  ***if pj > n then begin***  ***pj := n;***  ***inc(pi);***  ***end;***  ***j := pj;***  ***i := pi***  ***end;***  ***end;***  ***for i := 1 to N do***  ***begin***  ***for j := 1 to n do write(a[i,j]:6);***  ***writeln***  ***end;***  ***end.*** |

## 7. Визначення сторінок (В.О.Бардадим, В.В.Бондаренко)

При друкуванні великих документів може виникнути потреба друкувати не весь документ, а тільки деякі його сторінки. Серед аргументів програми друку є рядок з послідовністю номерів сторінок. Потрібно надрукувати не окремі сторінки, а діапазони сторінок і, можливо, вказувати початок і кінець діапазонів, а не послідовні числа.

*Завдання*: Напишіть програму, яка буде перетворювати списки сторінок у відповідну послідовність номерів сторінок.

*Вхідні дані*: Вхідний файл **PRІNT.DAT** містить один рядок, який має таку структуру: сторінка–1, сторінка–2, сторінка-3, ..., сторінка – ***N***.

Сторінка – ***і*** – або номер сторінки, або діапазон у вигляді початок–кінець (початок <= кінець).

Сторінки та діапазони перераховані в зростаючому порядку і не перетинаються. Діапазон включає початкову та кінцеву сторінки. Номери сторінок – числа від 1 до 1000000. 1 <= ***N*** <= 1000000.

*Вихідні дані*: Результат треба вивести до файлу **PRІNT.SOL** у вигляді сторінка-1, сторінка-2, сторінка-3,..., сторінка – ***М*** без пропусків.

*Технічні вимоги*: Ваша програма повинна мати назву **PRІNT.\***, де розширення залежить від мови програмування.

Приклад:

PRІNT.DAT

1,4-5,7-7,10-20

PRІNT.SOL

1,4,5,7,10,11,12,13,14,15,16,17,18,19,20

Будемо переглядати вхідний рядок і визначати фрагменти, що відокремлені комами. Якщо черговий фрагмент рядка складається з цифр, потрібно перетворити його на число і вивести до вихідного рядка. Якщо ж у фрагменті є дефіс, то його частини ліворуч і праворуч від дефіса перетворити на числа, що є початковим і кінцевим значенням циклу і у циклі вивести всі числа з утвореного діапазону до вихідного рядка.

***Program print;***

***var f : text;***

***poss, i, i1, ii : integer;***

***st : string;***

***function GetNum : integer;***

***var ss : string[10];***

***r, i : integer;***

***begin***

***ss := '';***

***while (poss <= length(st)) and ((st[poss] in ['0'..'9'])) do***

***begin***

***ss := ss + st[poss]; inc(poss);***

***end;***

***inc(poss); val(ss, r, i); GetNum := r***

***end;***

***begin***

***assign(f, ' print.dat'); reset(f); readln(f, st); close(f);***

***poss := 1;***

***repeat***

***i := GetNum;***

***if (poss <= length(st)) and (st[poss-1] = '-')***

***then i1 := GetNum else i1 := i;***

***for ii := i to i1 do***

***begin***

***write(ii);***

***if not ((ii=i1) and (poss >= length(st))) then write(',');***

***end;***

***until poss > length(st);***

***end.***

## 8. Мікроорганізми (М.З.Грузман)

Для обробки фотознімків мікроорганізмів, виконаних під мікроскопом, кожну фотографію розділено на дрібні клітинки. В кожній клітинці, яка повністю накрита одним з мікроорганізмів, або в якій міститься частина мікроорганізму, зроблено позначку.

Вважається, що дві клітинки з позначками належать одному й тому ж мікроорганізмові, якщо з однієї з них можна потрапити в іншу, рухаючись по клітинках з позначкою ліворуч, праворуч, вгору або вниз.

*Дано*: прямокутне фото розміром ***m*** x ***n*** клітинок, частину яких позначено.

*Отримати*: кількість організмів на фотознімку.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | \* | \* | \* |  |  |  |  |  |
|  | \* | \* | \* | \* | \* | \* | \* | \* |  |
| \* |  |  |  |  |  |  |  | \* |  |
|  |  |  |  | \* |  | \* |  | \* |  |
|  |  | \* |  | \* |  |  |  | \* |  |
|  | \* | \* |  | \* | \* | \* | \* | \* |  |

*Технічні вимоги*: Введення розмірів ***m*** та ***n*** на фотознімку здійснюється з клавіатури за запитом програми. Саме фото у закодованому вигляді міститься у файлі PHOTO.DAT. Кожному рядку клітинок відповідає запис у файлі. Позначеній клітинці відповідає символ "\*", а непозначеній – "0" (нуль).

Приклад. Фотознімок 6 х 10 має вигляд, зображений на рисунку. На цьому знімкові 4 мікроорганізми.

## 

|  |
| --- |
| Один з можливих розв'язкiв полягає в застосуваннi "в циклi" процедури, що реалiзує пошук в глибину для iдентифiкацiї позначених клiтин, що належать одному мiкроорганiзму.  Зазначимо, що iснує й ефективнiший однопрохiдний алгоритм.  ***uses crt;***  ***const maxm=100;***  ***maxn=100;***  ***hod : array[1..4,1..2] of integer = ((0,1),(0,-1),(1,0),(-1,0));***  ***var f : text;***  ***bact : array[1..maxm, 1..maxn] of integer;***  ***m, n, i, i1, x, y, col, k : integer;***  ***flStop, flF : boolean;***  ***st : string;***  ***begin***  ***fillchar(bact, sizeof(bact),0);***  ***assign(f, 'bactery.dat'); reset(f); readln(f, m, n);***  ***for i:=1 to m do***  ***begin***  ***readln(f,st);***  ***for i1:=1 to n do if st[i1]='\*' then bact[i,i1]:=1;***  ***end;***  ***close(f);***  ***col:=1;***  ***repeat***  ***flStop := true;***  ***for i:=1 to m do***  ***for i1:=1 to n do if (flStop) and (bact[i,i1]=1) then***  ***begin***  ***inc(col); bact[i,i1] := col; flStop := false;***  ***end;***  ***if flStop = false then***  ***repeat***  ***flF := true;***  ***for i:=1 to m do for i1:=1 to n do if bact[i,i1] = col then***  ***for k:=1 to 4 do***  ***begin***  ***x := i + hod[k,1]; y := i1 + hod[k,2];***  ***if (x>0)and(x<=m)and(y>0)and(y<=n)and(bact[x,y]=1) then***  ***begin bact[x,y] := col; flF := false end;***  ***end;***  ***until flF;***  ***until flStop;***  ***writeln('Col=',col-1);***  ***end.*** |

## 9. Дужки

Проаналізувати заданий текст з метою виявлення помилок у використанні дужок. Можливі три типи помилок:

а) невідповідність дужок ( і ) по кількості;

б) закриваюча дужка розміщена раніше відкриваючої;

в) відсутній зміст між дужками.

Результатом роботи програми повинно бути повідомлення про типи допущених помилок та їх місце в тексті (якщо це можливо).

|  |
| --- |
| Під час проходження по всьому рядку перевіряємо кількість відкритих і закритих дужок і при виявленні помилки, вказаної в умові, виводимо відповідне повідомлення. Головне – вірно враховувати відповідність відкритих і закритих дужок.  ***program braces;***  ***var st : string;***  ***i, c, c1 : integer;***  ***begin***  ***write('Введіть рядок: '); readln(st);***  ***c := 0; c1 := 0;***  ***for i := 1 to length(st) do***  ***case st[i] of***  ***'(' : begin***  ***inc(c); inc(c1);***  ***if (i = length(st)) or (st[i+1] = ')') then***  ***writeln('Помилка: знайдено "()". Pos: ',i)***  ***end;***  ***')': begin***  ***dec(c1);***  ***if c-1<0 then***  ***writeln('Помилка: знайдено ")" без "(". Pos: ',i)***  ***else dec(c)***  ***end;***  ***end;***  ***if c1<>0 then***  ***writeln('Помилка: невiдповiднiсть по кiлькостi дужок "(" та ")"')***  ***end.*** |

## 10. Греко–латинський квадрат

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 4 | 3 | 2 | 1 |
| 2 | 1 | 4 | 3 |
| 3 | 4 | 1 | 2 |

Греко–латинським квадратом називається квадрат ***N*** x ***N***, в кожному рядку, в кожному стовпці і в кожній діагоналі якого містяться всі цілі числа від 1 до ***N***. Приклад такого квадрата 4 х 4.

Написати програму, яка:

1. будує хоча б один квадрат порядку ***N***;
2. будує всі квадрати порядку ***N***;
3. будує всі квадрати порядку ***N*** так, що не можна отримати один з іншого при допомозі поворотів і обертань навколо осей симетрії.

|  |
| --- |
| Греко–латинських квадратів розміром 2 х 2 та 3 х 3 не існує. Один з квадратів розміром 4 х 4 приведено в умові задачі.  Задача розв'язується повним перебором, оскільки поки що невідомо іншого способу [7]. З метою спрощення повного перебору слід використати лексикографічний метод генерації перестановок ***n*** чисел у кожному рядку таблиці (він описаний в задачі Чотиризначні числа). Звертаємо увагу на той факт, що, можливо, і швидше всього – обов'язково, нам буде потрібно на деякому етапі повернутись до попереднього рядка таблиці і вибрати наступну перестановку. Тому дана задача є повчальною з цієї точки зору одразу для засвоєння двох методів: бектрекінгу і лексикографічного методу генерації перестановок. Саме тому, крім вказівки на саму ідею розв'язання, ми не робимо жодних інших підказувань і рекомендуємо самостійно розв'язати дану задачу. Її розв'язок – ключ до успіху на майбутніх олімпіадах! |