Министерство Образования и Науки Российской Федерации  
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ

**Курсовая работа**

**по практикуму на ЭВМ: структуры данных и алгоритмы**

Факультет: прикладной математики и информатики

Группа: ПМ-53

Студент: Тябин Егор Алексеевич

Преподаватель: Тракимус Юрий Викторович

### Новосибирск 2016

# Условие задачи

Подсчитать количество попарно неизоморфных графов с n вершинами и четырьмя ребрами.

# 2. Анализ задачи

2.1. Исходные данные задачи: *n* – количество вершин, *n∈N.*  
2.2. Результат: *result ∈ N –* количество попарно не изоморфных графов.

2.3. Решение.   
Для начала определим, какие графы будут использоваться в нашей задаче. Воспользуемся определением графа.

*Граф* - это пара *(V, E),* где V-конечное непустое множество вершин, а E – множество неупорядоченных пар ˂u, w˃ вершин из V, называемых ребрами. То есть граф — это множество точек и множество линий, соединяющих эти точки. Ориентированный граф, или орграф, G= (V, E) отличается от графа тем, что E - это множество упорядоченных пар (u, w) вершин, u, w∈V, называемых дугами. Дуга (u, w) ведет от вершины u к вершине w. При этом вершину w называют преемником вершины u, а u – предшественником вершины w.

*Граф* есть некоторое множество вершин и некоторое множество ребер, соединяющих пары различных вершин (одно ребро может соединять максимум одну пару вершин).

**Граф** - Граф G состоит из конечного непустого множества V, содержащего р вершин, и заданного множества Х, содержащего q неупорядоченных пар различных вершин из V.  
….Отметим, что из определения вытекает, что в графе не может быть петель, т.е. ребер, соединяющих вершины сами с собой…

**Мультиграф** – в мультиграфе не допускаются петли, но пары вершин могут быть соединены более чем 1 ребром (кратные ребра).

**Псевдограф** – мультиграф, в котором допускаются петли.

*Графом G (V, E)* называется совокупность двух множеств – непустого множества V (множества вершин) и множества E неупорядоченных пар различных элементов множества V (E – множество ребер).

### G(V, E) = <V; E>, V≠[Ø](https://ru.wikipedia.org/wiki/%C3%98), E<V x V, E=E-1.

Воспользуемся свойством графа из книги Р. Седжвика «Фундаментальные алгоритмы на с++». *Граф,* состоящий из V вершин, содержит не более V(V-1)/2 ребер.  
Исходя из этого свойства определим, сколько минимум вершин может содержать наш граф. 2\*(2-1)/2=1 ребро, 3\*(3-1)/2=3 ребра, 4\*(4-1)/2=6 ребер.   
Таким образом наш граф может иметь минимум 4 вершины.  
Для решения задачи нам понадобится определение изоморфного графа.

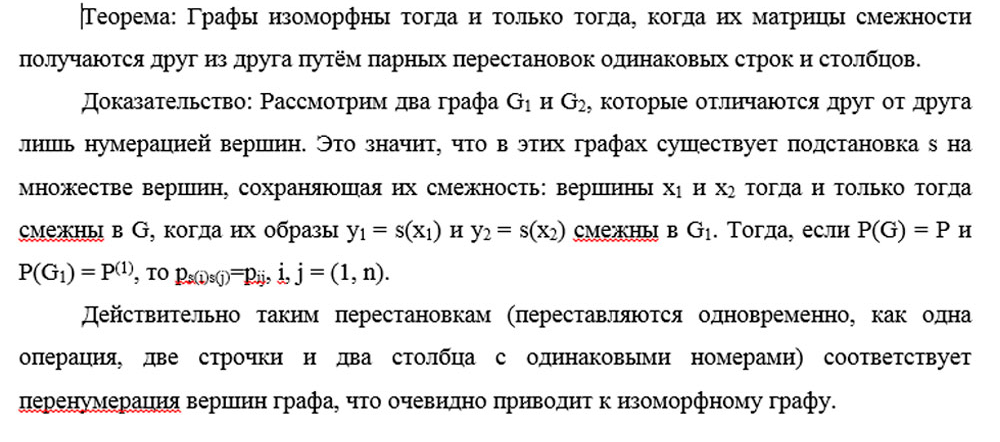
Два графа называют *изоморфными*, если можно поменять метки вершин на одном из них таким образом, чтобы набор ребер этого графа стал идентичным набору ребер другого графа.

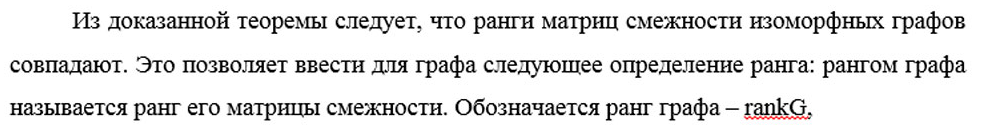
Говорят, что два графа G1(V1, E1) и G2(V2, E2) *изоморфны* (обозначается G1 ~ G2), если существует биекция h: V1→ V2, сохраняющая смежность:

E1=(u, v) \in E1 →e2=(h(u), h(v)) \in E2,

E2=(u, v) \in E2 →e1=(h-1(u), h-1(v)) \in E1,

Два графа *G*1 и *G*2 *изоморфны*, если существует взаимно однозначное отображение множества вершин графа *G*1 на множество вершин графа *G*2, сохраняющее смежность.





По условию задачи надо посчитать максимально возможное количество не изоморфных между собой графов с n вершинами и четырьмя ребрами, а затем подсчитать количество попарно неизоморфных графов.  
Попарная неизоморфность означает, пару графов которые не изоморфны между собой.  
Значит количество попарных неизоморфных графов означает количество максимально возможных таких пар.

# 3. Программа

#include <iostream>

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <iomanip>

using namespace std;

int isomorphic (int \*\*c, int \*\*v, int M){

int i, j, q, l, s, d;

bool isom;

int \*\*g = new int\*[M];

for (i = 0; i < M; i++){

g[i] = new int[M];

}

int\*\* o = new int\*[M];

for (i = 0; i < M; i++){

o[i] = new int[M];

}

for (i = 0; i < M - 1; i++){

q = i+1;

for (j = q; j < M; j++){

isom = true;

for (l = 0; l < M; l++){

for (d= 0; d < M; d++){

o[l][d] = c[l][d];

}

}

for (l = 0; l < M; l++){

o[l][j] = c[l][i];

o[l][i] = c[l][j];

}

for (s = 0; s < M; s++){

for (d = 0; d< M; d++){

g[s][d] = o[s][d];

}

}

for (l = 0; l < M; l++){

g[i][l] =o[j][l];

g[j][l] = o[i][l];

}

for (s = 0; s < M; s++){

for (d = 0; d < M; d++){

if (g[s][d] != v[s][d]) isom = false;

}

}

if (isom == true) return 1;

}

}

return 0;

}

void main(){

setlocale(LC\_CTYPE, "Russian");

int q, j, i, k, N, w, e, r, t, y, u, z, x, notiz, answer=0;

int g[4][2];

printf("Введите количество вершин: ");

scanf\_s("%d", &N);

int n = ((N\*(N - 1) / 2)\*((N\*(N - 1) / 2) - 1)\*(N\*(N - 1) / 2 - 2)\*(N\*(N - 1) / 2 - 3)) / (2 \* 3 \* 4);

int \*\*\*b = new int\*\* [n];

for (i = 0; i < n; i++){

b[i] = new int\* [N];

}

for (i = 0; i < n; i++){

for (j = 0; j < N; j++){

b[i][j] = new int[N];

}

}

int\*\*\* a = new int\*\*[n];

for (i = 0; i < n; i++){

a[i] = new int\*[4];

}

for (i = 0; i < n; i++){

for (j = 0; j < 4; j++){

a[i][j] = new int[2];

}

}

k = 0;

for (q = 1; q <= N; q++){

g[0][0] = q;

for (w = 1; w <= N; w++){

g[0][1] = w;

for (e = 1; e <= N; e++){

g[1][0] = e;

for (r = 1; r <= N; r++){

g[1][1] = r;

for (t = 1; t <= N; t++){

g[2][0] = t;

for (y = 1; y <= N; y++){

g[2][1] = y;

for (u = 1; u <= N; u++){

g[3][0] = u;

for (i = 1; i <= N; i++){

g[3][1] = i;

notiz = 0;

for (j = 0; j < 4; j++){

if (g[j][0] >= g[j][1]) notiz =1;

}

if (!notiz){

for (j = 0; j < 3; j++){

z = g[j][0] \* 10 + g[j][1];

x = g[j + 1][0] \* 10 + g[j + 1][1];

if (z >= x) notiz = 1;

}

}

if (!notiz){

a[k][3][1] = g[3][1];

a[k][3][0] = g[3][0];

a[k][2][1] = g[2][1];

a[k][2][0] = g[2][0];

a[k][1][1] = g[1][1];

a[k][1][0] = g[1][0];

a[k][0][1] = g[0][1];

a[k][0][0] = g[0][0];

k++;

}

}

}

}

}

}

}

}

}

for (i = 0; i < n; i++){

for (q = 0; q < N; q++){

for (w = 0; w < N; w++){

b[i][q][w] = 0;

}

}

}

for (i = 0; i < k; i++){

for (q = 0; q < N; q++){

b[i][a[i][q][0]-1][a[i][q][1]-1] = 1;

b[i][a[i][q][1]-1][a[i][q][0]-1] = 1;

}

}

q = 0;

while (q < k-1){

for (i = q+1; i < k; i++){

if (isomorphic(b[q], b[i], N)) answer++;

}

q++;

}

for (i = 0; i < n; i++){

for (j = 0; j < N; j++){

for (q = 0; q < N; q++){

printf("%d ", b[i][j][q]);

}

printf("\n");

}

printf("\n");

}

\_getch();

}