**Отчёт по расчетной работе**

**Тема:** Графы

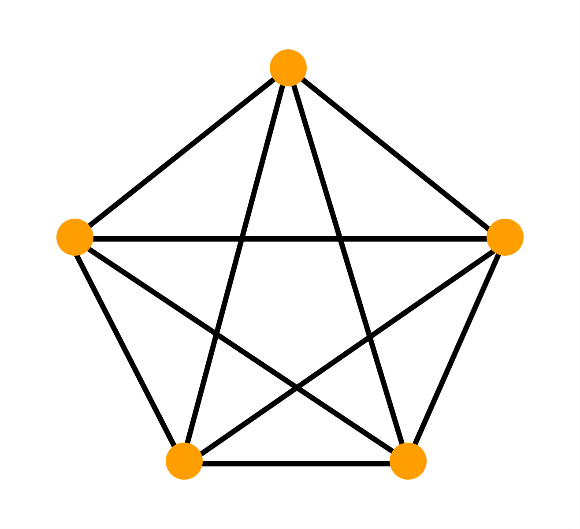
**Цель:** Опредлить,является ли вводимый граф-графом Паппа

**Задача:** Создать программу, которая будет брать данные из файла и по матрице инцидентности будет проверять , является ли граф- графом Паппа

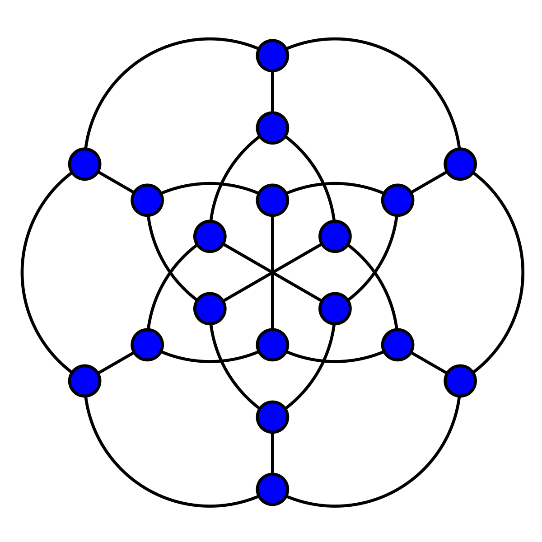
**Вариант** 1.20(ми)

**Список ключевых понятий (определения):**

Граф- это совокупность непустого [множества](https://dic.academic.ru/dic.nsf/ruwiki/4759) вершин и множества пар вершин (связей между вершинами).



Граф Паппа -  [двудольный](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D1%83%D0%B4%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84) 3-[регулярный](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84) неориентированный граф с 18 вершинами и 27 рёбрами.

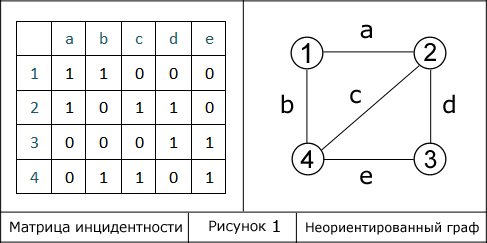


**Матрица инцидентности**

**Инцидентность** – понятие, используемое только в отношении ребра и вершины: две вершины (или два ребра) инцидентными быть не могут.

Матрица состоит из n строк которое равно числу вершин графа, и m столбцов, которое равно числу ребер. Таким образом полная матрица имеет размерность n x m. То есть она может быть, как квадратной, так и отличной от нее.

Ячейка в позиции I (i, j) равна 1 тогда, когда вершина инцидентна ребру иначе мы записываем в ячейку 0.



**Алгоритм:**

1.Выбрать файл с матрицей;

2.Посчитать количество вершин и ребер;

3.Создать матрицу Nхn ,где N -количество вершин, n-количество ребер

4.Проверить на кол-во вершин и ребер

5. Проверить чтобы каждая вершина имела 3 связи

6.Вывести матрицу инцидентности

7.Вывести является ли исходный граф графу Паппа.

**Код:**

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

int num = 0;

string nameOfFile = "";

cout << "Введите номер файла, где хранится граф: ";

cin >> num;

switch (num)

{

case 1:

nameOfFile = "D:\\расч\\graph.txt";

break;

case 2:

nameOfFile = "D:\\i.txt";

break;

case 3:

nameOfFile = "D:\\grap.txt";

break;

case 4:

nameOfFile = "D:\\k.txt";

break;

case 5:

nameOfFile = "D:\\h.txt";

break;

default:

break;

}

ifstream fin(nameOfFile);

char first = '0';

int n = 1;

int N = 0;

// Подсчет строк и столбцов

while (fin.get(first)) {

if (first == ' ' && N == 0) n++;

if (first == '\n') N++;

}

fin.close();

// Если последний символ в файле не является символом новой строки, увеличиваем счетчик строк

if (first != '\n' && N > 0) {

N++;

}

cout << "Кол-во строк: " << N << endl;

cout << "Кол-во столбцов: " << n << endl;

//Создание матрицы и чтение из файла

fin.open(nameOfFile);

int\*\* matrix = new int\* [N];

for (int i = 0; i < N; i++)

{

matrix[i] = new int[n];

}

for (int i = 0; i < N; i++)

{

for (int j = 0; j < n; j++)

{

fin >> matrix[i][j];

}

}

cout << '\n';

bool isGraph = true;

//Проверка на кол-во вершин

if (N != 18 || n != 27) isGraph = false;

for (int i = 0; i < N; i++)

{

int count = 0;

for (int j = 0; j < n; j++)

{

if (matrix[i][j] == 1)

{

count++;

}

}

if (count != 3) isGraph = false;

}

for (int i = 0; i < N; i++)

{

for (int j = 0; j < n; j++)

{

cout << matrix[i][j];

cout << ' ';

}

cout << '\n';

}

cout << '\n';

if (isGraph) cout << "Этот граф является графом Паппа";

else cout << "Этот граф не является графом Паппа";

fin.close();

for (int i = 0; i < N; i++)

{

delete[] matrix[i];

}

delete[] matrix;

return 0;

}