Министерство образования Республики Беларусь

Учреждение образования

“Брестский Государственный технический университет”

Кафедра ИИТ

Лабораторная работа № 2

По дисциплине «Математические основы И С»

Тема: Нахождение эйлеровых и гамильтоновых циклов в неориентированном графе.

Выполнил:

Студент 2 курса

Группы ИИ-23

Макаревич Н. Р.

Проверил:

Козинский А. А.

Брест 2023

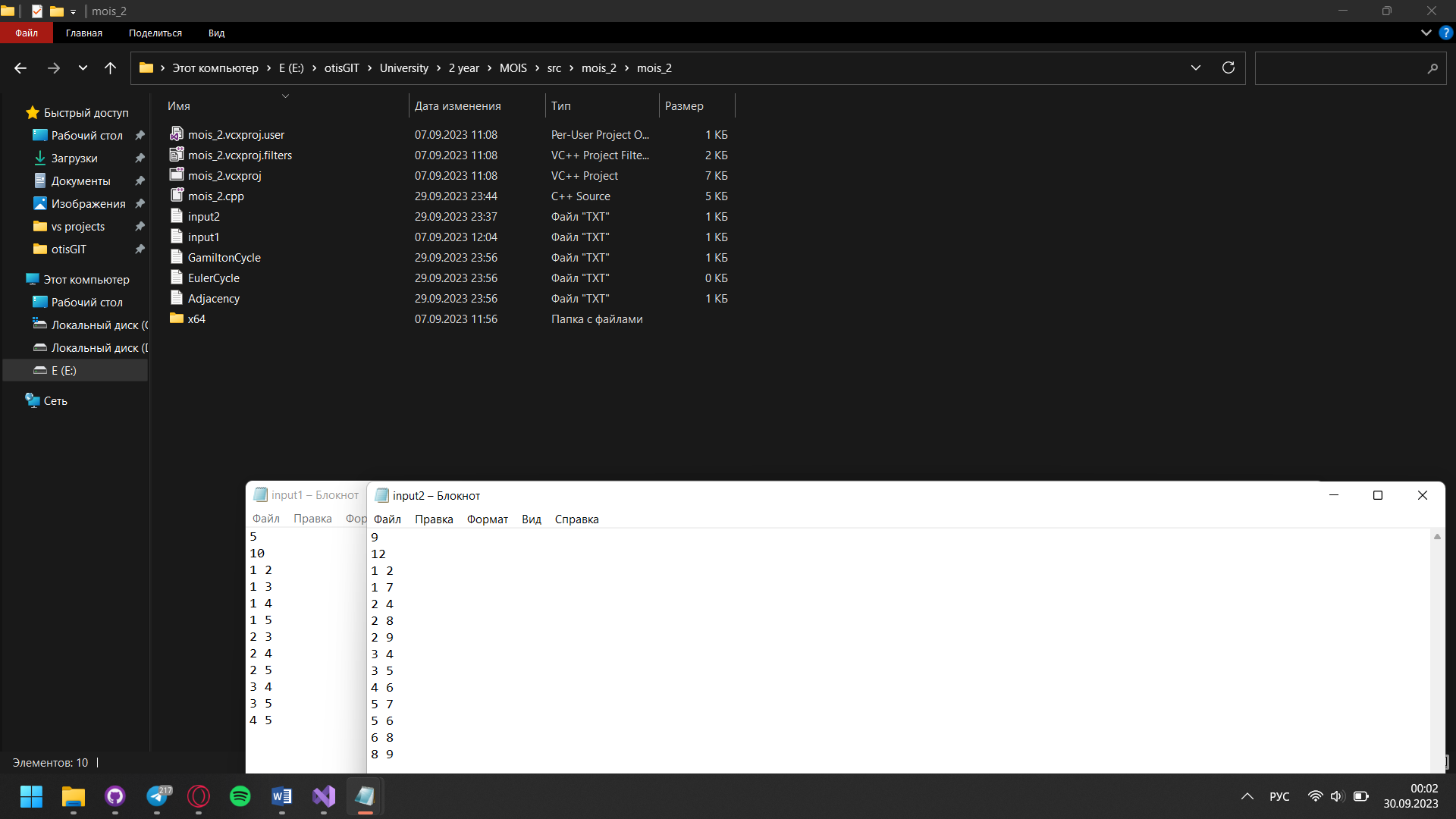
Вариант 3

***Задание.***

1. Написать программу нахождения эйлерова цикла в графе. Результатом программы является последовательность вершин в цикле. Предусмотреть проверку критерия на наличия эйлерова цикла в графе.
2. Написать программу нахождения гамильтонова цикла в графе. Результат программы – последовательность вершин цикла.
3. Варианты заданий указаны в таблице. Графы заданы списком ребер, *n* – количество вершин, -- проверка графа на эйлеров цикл,  - на гамильтонов цикл.
4. Изобразить графы и циклы в них.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Граф | *n* | Граф | *n* |
| 3. | (1,2),(1,3),(1,4),(1,5),(2,3),(2,4),  (2,5),(3,4),(3,5),(4,5) | 5 | (a,b),(a,g),(b,d),(b,h),(b,i),(c,d),(c,e)  (d,f),(e,g),(e,f),(f,h),(h,i) | 9 |

Файл с входными данными input.txt:



Код программы:

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

void showVector(vector<int> v, string name) {

ofstream fout;

fout.open(name);

for (int i = 0; i < v.size(); i++)

fout << v[i] + 1 << " ";

fout.close();

}

void showMatrix(int\*\* matrix, int n) {

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

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

cout << matrix[i][j] << " ";

}

cout << endl;

}

}

class Graph {

public:

int\*\* matrix;

int numOfVertex, numOfEdge;

Graph(string name) {

this->name = name;

ifstream fin;

fin.open(name);

fin >> numOfVertex >> numOfEdge;

fin.close();

}

void createAdjacencyMatrix() {

ifstream fin;

fin.open(name);

matrix = new int\* [numOfVertex];

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

matrix[i] = new int[numOfVertex];

for (int i = 0; i < numOfVertex; i++) {

for (int j = 0; j < numOfVertex; j++) {

matrix[i][j] = 0;

}

}

int trash;

fin >> trash >> trash;

for (int i = 0; i < numOfEdge; i++) {

int from, to;

fin >> from >> to;

matrix[from - 1][to - 1] = 1;

matrix[to - 1][from - 1] = 1;

}

ofstream fout;

fout.open("Adjacency.txt");

fout << "\t";

for (int i = 1; i <= numOfVertex; i++)

fout << i << "\t";

fout << endl;

for (int i = 0; i < numOfVertex; i++) {

fout << i + 1 << "\t";

for (int j = 0; j < numOfVertex; j++) {

fout << matrix[i][j] << "\t";

}

fout << endl;

}

fout.close();

fin.close();

cout << "complited adj" << endl;

}

vector <int> eulerCycle(int start) {

if (checkForOdd()) {

int\*\* matrixTemp = matrix;

start = 0;

vector <int> temp;

vector <int> res;

temp.push\_back(start);

cout << "starting euler" << endl;

while (!temp.empty()) {

for (int j = 0; j < numOfVertex; j++) {

if (matrixTemp[start][j] == 1) {

temp.push\_back(j);

matrixTemp[start][j] = 2;

matrixTemp[j][start] = 2;

start = j;

break;

}

if (j == numOfVertex - 1) {

res.push\_back(temp.back());

temp.pop\_back();

if (!temp.empty())

start = temp.back();

}

}

}

cout << "ending euler" << endl;

return res;

}

cout << "This graph have no Euler cycle" << endl;

vector <int> null;

return null;

}

vector <int> gamiltonCycle(int v0) {

cout << "starting gamilton" << endl;

vector <int> path(numOfVertex);

path[0] = v0;

vector <bool> isVisited(numOfVertex, false);

isVisited[v0] = true;

if (gamilton(path, isVisited, 1, v0)) {

cout << "ending gamilton" << endl;

return path;

}

else cout << "ERROR; ending gamilton" << endl;

}

private:

string name;

bool checkForOdd() {

int counter = 0;

for (int i = 0; i < numOfVertex; i++) {

for (int j = 0; j < numOfVertex; j++) {

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

counter++;

}

if (counter % 2 == 1)

return false;

}

return true;

}

bool gamilton(vector <int>& path, vector <bool>& isVisited, int k, int v0)

{

bool res = false;

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

{

if (matrix[i][path[k - 1]])

{

if (k == numOfVertex && i == v0)

res = true;

else

if (!isVisited[i])

{

isVisited[i] = k;

path[k] = i;

res = gamilton(path, isVisited, k + 1, v0);

if (!res) isVisited[i] = false;

}

}

else continue;

}

return res;

}

};

int main()

{

cout << "Enter name of the file: ";

string name;

cin >> name;

Graph graph(name);

graph.createAdjacencyMatrix();

cout << "Enter staring vertex: ";

int start;

cin >> start;

vector <int> res = graph.eulerCycle(start);

showVector(res, "EulerCycle.txt");

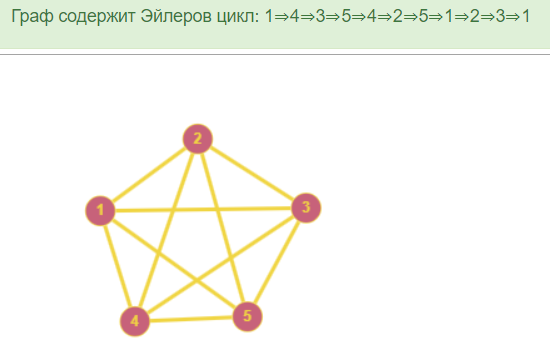
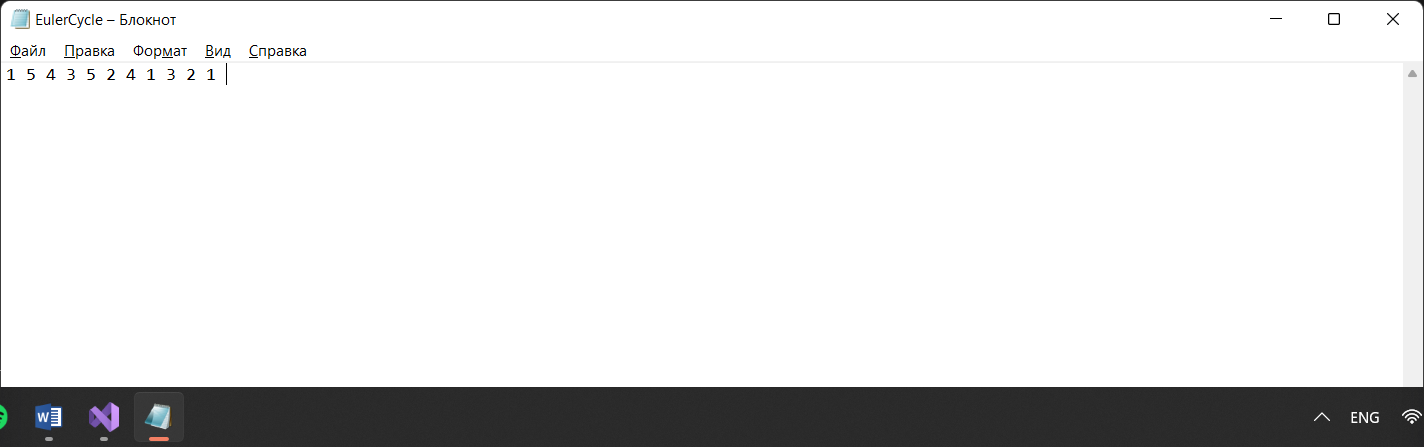
res = graph.gamiltonCycle(start);

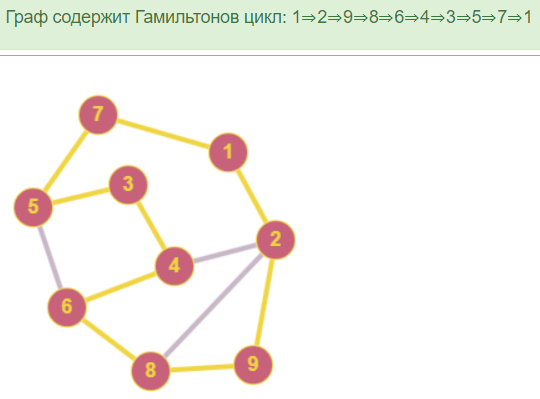
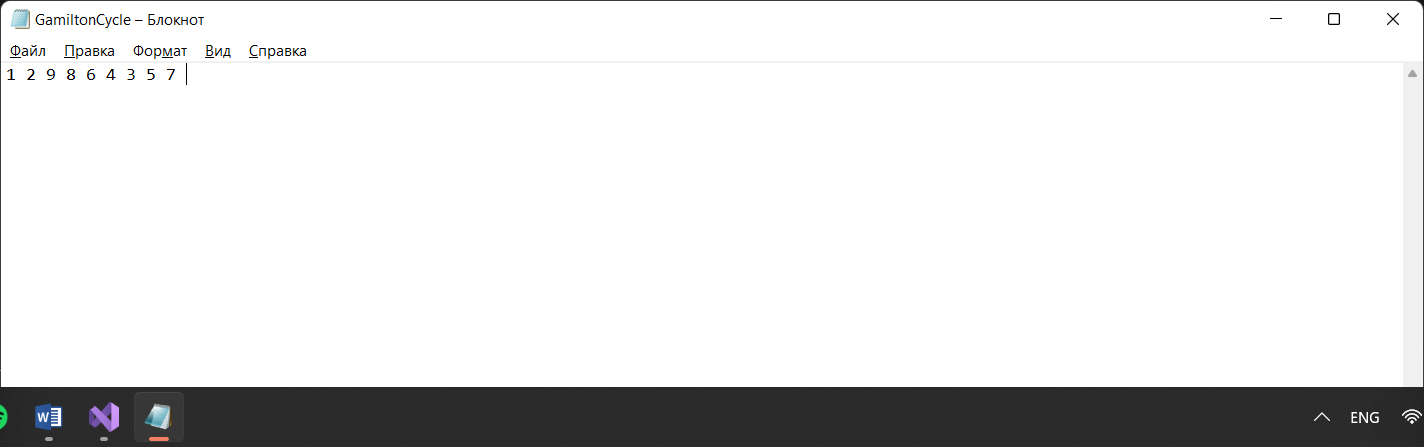
showVector(res, "GamiltonCycle.txt");

return 0;

}

Вывод программы:





Вывод: Написал программу нахождения Эйлерова и Гамильтонова циклов.